GSM Remote Control System

Micro computer system

Project group 415
E4 - project, 2008
IES

AALBORG UNIVERSITET

AALBORG UNIVERSITET

Department of Electronic Systems
Fredrik Bajers Vej 7B
9220 Aalborg Ø
Denmark
Phone +45 96 35 98 36
Fax +45 98 15 36 62
http://es.aau.dk

Title: GSM Remote Control System

Subject: Micro computer Systems

Project period: 4/2 – 29/5, 2008

Group: E4 415

Group members:
Simon Konge
Kristian Kiib Gregersen
Lars Jespersen
Rasmus Nielsen
Henrik Christoffersen

Technical adviser:
Persefoni Kyritsi

Synopsis:

This report deals with the development and design of a remote control system, that utilize GSM technology. The system is designed to manage the control and monitoring of a small heating system. The design of the system allows implementation of further applications, but the detail of these will remain unspecified.
The system is designed around the Motorola 68000 microprocessor, that make up the hart of the system. The system features both local and remote user interface. The local interface consists of a numeric keypad and a LCD. The remote interface consists of a GSM module, that allow the administrator/user to communicate with the system using text messages.
The system is operational and functional in accordance with the performance specifications. The system software is implemented "by proxy" using the TS2 Monitor debugging device, that allow the system to be run from the system RAM instead of burning the system on the system ROM.

Editions: 7 ed.
Pages: 166
Attachments: 1 CD attached
Completion of project: 29/5-2008

The contents of this report is freely available, however, publication with source reference may only occur as agreed with the authors.

Chapter 1

Preface

This project is presented by group 08gr415 from the Institute of Electronic Systems at Aalborg University - AAU, on the subject *Microcomputer Systems*. The report is written for readers, who has a technical understanding equal to or greater than 4^{th} semester at AAU.

The semester has the purpose of giving the students an understanding of Microcomputer Systems, including architecture, structure, interfacing, and programming. To do this a series of project suggestions were available for the students to work with. The one chosen for this project is the development of a GSM Remote Control System for holiday homes.

The bibliography can be found on page 105, and a list of abbreviations used in this report, can be found on 106. When words are overlined, such as $\overline{LDS}$ and $\overline{UDS}$ it means, that the function is active low.

The group would like to thank Associate Professor Sofus Birkedal Nielsen and Assistant Engineer Claus Walther, for their guidance in troubleshooting the GSM Remote Control System. During the design and implementation process, the laboratory in B1-101 has been used.

Kristian Gregersen

Lars Jespersen

Simon Konge Koldbæk

Henrik Christoffersen

Rasmus Birkelund Nielsen

Contents

1 Preface **5**

2 Introduction **11**

3 System Analysis **12**

3.1 Project structure procedure 12

3.2 System overview 13

3.3 System specification 14

3.4 Use cases 15

3.5 Performance specifications 23

3.6 Performance verification 26

3.7 User manual 28

4 Hardware Design **35**

4.1 The minimal system 35

4.2 Memory and Addressing 40

4.3 PEEL 43

4.4 GSM module 43

4.5 RS-232 Connection 45

4.6 Temperature Sensor and ADC 49

4.7 Liquid Crystal Display 52

4.8 Keypad 57

4.9 Mock system 59

4.10 Address Decoding 61

4.11 Microprocessor Communication 62

4.12 Implementation 64

5 Software Design **67**

5.1 Design procedure 67

5.2 Program structure 68

5.3 Storage structure 70

5.4 TS2 Monitor 73

5.5 Program design 73

5.6 GSM software 76

5.7 Local interface software 84

5.8 ADC software 88

5.9 Alarm update 90

5.10 Terminate user 91

5.11 Control Update 92

5.12 Implementation 93

6 Accept tests **95**

6.1 I/O device test results 95

6.2 Communication test results 97

6.3 System test results 98

6.4 Summary 100

7 Conclusion **101**

8 Future plans **103**

Bibliography **105**

A List of abbreviations and terms **106**

B Testjournal - I/O devices **108**

B.1 Preface 108

B.2 Equipment 108

B.3 Setup 108

B.4 Tests 109

C Testjournal - Communication **111**

C.1 Preface . . . 111
C.2 Equipment . . . 111
C.3 Setup . . . 111
C.4 Tests . . . 112

D Testjournal - System **114**
D.1 Preface . . . 114
D.2 Equipment . . . 114
D.3 Setup . . . 114
D.4 Tests . . . 115

E Questionnaire - Intuitivity test of GSM RCS **116**
E.1 Introduction . . . 116
E.2 Participant information . . . 116
E.3 Questions . . . 116

F Complete schematic **118**

G PEEL software **119**
G.1 PEEL 1 . . . 119
G.2 PEEL 2 . . . 120
G.3 PEEL 3 . . . 120

H GSM software **122**
H.1 Main Functions . . . 122
H.2 Subroutines . . . 125
H.3 Constants . . . 133

I LCD software **136**
I.1 Main Functions . . . 136
I.2 Subroutines . . . 156

J ADC and Temperature software **161**

K Alarm software **163**

L User termination software **165**

M Control unit software **166**

Chapter 2

Introduction

For many people, a holiday home is a popular way to spend the holidays and weekends. There are also people, that use their holiday home as an ”investment” and rent it to tourists. The owners permanent residence may be far from the holiday home, making it time-consuming to inspect the state of the house. A remote control and monitoring system, would make it easier to inspect the holiday home and make sure that everything is in order.
A practical remote control and monitoring system for a holiday home could be a system, that controls and monitors temperature levels, i.e. the indoor temperature. If the house is equipped with a swimming pool, it would be extremely practical to be able to control and monitor the temperature in the pool, because of the time, it takes to alter the temperature in a pool. The device would enable the owners to monitor and control temperature levels in their holiday home from any where. In this manner, the temperatures in the holiday home could be set to preferred levels before arrival.
A logical improvement of the system would be the possibility to monitor and control other parameters such as door locks and movement sensors. This would be a huge help for the owner of a rental holiday home. The system would make it easy to check whether tenants, after their visit, remember to lock the doors and turn off the heat. If not, it would be convenient to be able to do so remotely. However logical, a wider range of control and monitoring will not alter the basic functionality of the system. The system will therefore be built solely as a means of controlling and monitoring the temperature. This will ease the implementation of other subsystems in a potential improvement of the system.

Chapter 3

System Analysis

This project deals with the development and design of a GSM RCS (**G**lobal **S**ystem for **M**obile communications **R**emote **C**ontrolled **S**ystem). The system uses an Motorola 68000 MPU (Micro Processor Unit) as a basis. The MPU will be used to enable remote monitoring and control of electrical devices and support local monitoring and control of these electrical devices. In order to communicate with the system remotely, a GSM module is implemented.

3.1 Project structure procedure

The project structure is based on the SPU model that is often used in relation to the development of new products. The model involves organizing the work process in accordance to the size of the individual subtasks. The project follows the structure illustrated in figure 3.1. The v-shape indicates the size, and the time reserved for the different tasks. It is noted that the model also illustrates the connection between the different tasks. The left side of the figure contains the tasks and the right side contains a matching ending/evaluation of the tasks. The project structure and

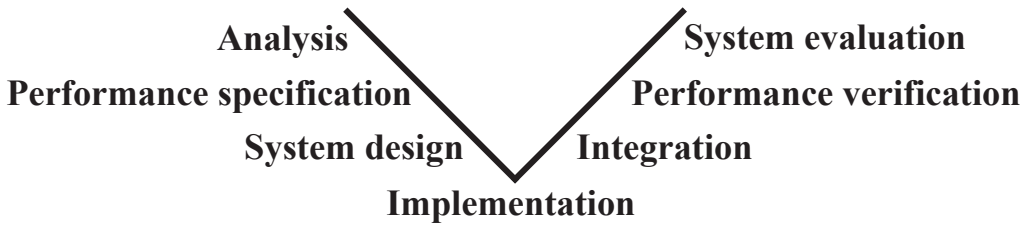

Figure 3.1: V arranged SPU model describing the project work process

process is based on the SPU model but the actual report is divided into two separate stages, the design and implementation of the hardware and the design and implementation of the software. I.e. two v-shaped processes are used, one related to the hardware and one related to the software. This duality is not illustrated on figure 3.1 as this only deals with the general structure of the project.

The SPU model is normally used when a product is developed in collaboration with a company. Though this project does not have this resource, the model is a good tool in relation to planning the project and the tasks related to its development. The project's lack of a company partner also entails, that the analysis is based primarily on different use cases, which describe the usage of the product.

3.2 System overview

In this section the general functionality of the system will be outlined.
Because the system must handle large amounts of data and function autonomously, when not controlled by the administrator/user, an MPU is implemented to manage the operation of the system. The MPU also enables the implementation of various system features solely through software, thereby simplifying the overall design of the system.
The remote communication is achieved using the GSM network, that is normally used for mobile phone communication. The group intends to employ a GSM module capable of transmitting and receiving text messages. These messages form the basis of the remote communication.
In order to enable the user to control and monitor the system locally, a local user interface is implemented. The local user interface consists of an LCD (Liquid Crystal Display) and a keypad. The reason that an LCD and a keypad are used is that these very versatile elements allows for implementation of further system features without changing the complexity of the user interface.
The designed system will only be able to control a small mock system and not an actual electrical device in the common sense. The mock system consists only of a controllable digital switch capable of changing between the two logic states high and low. This signal is used to control a small LED (Light Emitting Diode). The mock system could in reality be several devices with different functionality. In the report the mock system, for consistency, will be regarded as a switch that controls an electrical heater.

The main focus of the project is five specific features that, in this project, are illustrated in relation to the control and monitoring of the temperature in a holiday home. The specific features that are explained in the project are listed below.

- Remote control of an electrical device.
- Remote monitoring of an electrical device.
- Autonomous monitoring of an electrical device.
- Autonomous control of an electrical device.
- Remote alarming in relation to the monitoring of an electrical device.

In this project the GSM RCS is used as a means to control the mock system i.e. an electrical heater. This primary function is accompanied by a series of monitoring and alarm features. All the features in the system relate to the control and monitoring of temperature, but the same basic procedure could be employed to enable control and monitoring of any electrical device.

The five features are mapped into this project in accordance with the list below that state the specific tasks that the system will be able to manage.

- Remote control of the electrical heater using GSM.
- Remote monitoring of the ambient temperature using GSM.
- Continuous monitoring of the ambient temperature. (Software based thermostat)
- Continuous control of the electrical heater. (Software based thermostat)
- Remote alarming using GSM in relation to the monitoring of the ambient temperature

3.3 System specification

This section describes the administrator/user-system communication. The objective of this section is to identify the different system boxes, needed to obtain the desired functionality.
As mentioned earlier the system is able to handle two types of communication, remote and local.

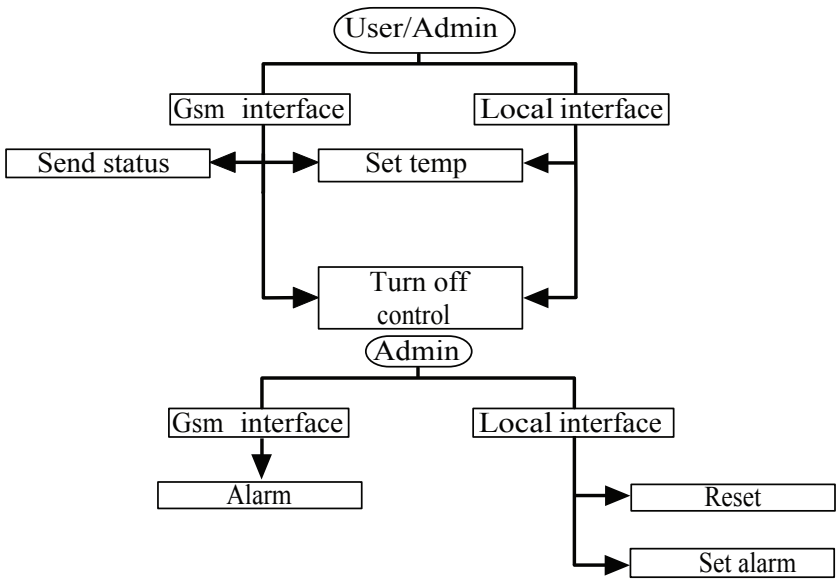

Figure 3.2: System user interfacing

As shown in figure 3.2 the user will not have the option to request a status report locally as this will be displayed at all times. The alarm function is only available remotely, because it is meant as an alarm in an extreme situation e.g. if the house is on fire or the temperature becomes very low indicating the need for winter preparations. The alarm temperature can however be configured locally.
By having determined the primary functions of the systems, it is reasonable to determine which system blocks are required. Based on its versatility, it is decided to use an MPU based system to handle the functionality of the system through programming. In order to enable the system to manage the different functions, a series of I/O devices must be implemented. The purpose of

these I/O devices is to enable the system to receive/send information related to the different tasks. The following is a description of the I/O devices that are to be implemented and used, in order to obtain the desired functionality.

3.3.1 GSM module

In order to enable the system to be accessed remotely by the user and the system to remotely alert the user a GSM-module is implemented. When not actively used by the system the GSM module is in standby mode ready to receive information from the user. When a message is received, the GSM-module must be able to interrupt the system, as the user commands would have first priority in the system.

3.3.2 Temperature sensor

The temperature sensor will be designed so it can be placed at a distance from the main systems physical placement. Furthermore the amount of electrical circuits outside the main system must be kept at a minimum.

3.3.3 Temperature control/Mock system

In this project there will not be made a specific temperature control system. The actual control consists of a mock system capable of controlling a small LED, that indicates whether the heater is on or off.

3.3.4 Display

The display is the local means to communicate information from the system to the user. When the display is not used for a specific task, it will show the current temperature, the status of the control unit, the time and the status of the GSM module.

3.3.5 Keypad

The keypad is the user's primary way to setup and control the system. All commands through this interface will be valid. The commands, delivered through the GSM module by the means of incoming text messages, undergo system authentication and a syntax check.

3.4 Use cases

In order to establish the functionality of the GSM RCS under normal use, a series of use cases are described. The purpose of these use cases is to identify and specify concrete performance specification demands. The use cases describe the normal use of the system and double as a source of inspiration for the implemented software. This duality is possible because the structure of the

use cases is close to that of the implemented software, though not all functions are represented.

The use cases are purely based on the product's fundamental features, and do not describe the internal functions of the system.
The product can be described using the model illustrated in figure 3.3. There are three different participants in this model: the "administrator/user", the "temperature sensor" and the "control unit". These participants can influence the system in different ways, in accordance to the functionality of the system. The model seeks to illustrate the different ways these participants can interact.
Note that the "system-box" is a fictitious object, which only illustrates the internal structure of the system. The "system-box" consists of various elements, that manage the communication between the participants.

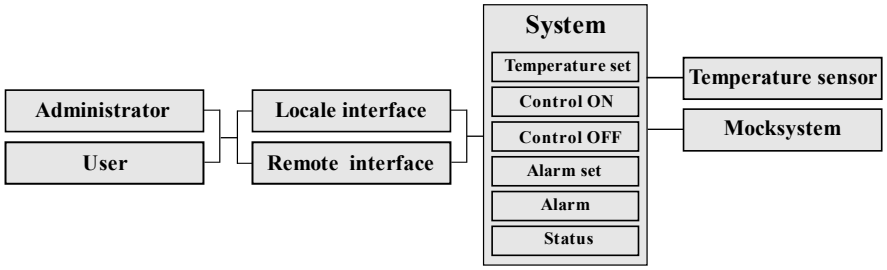

Figure 3.3: Model describing the participants in the product

The systems user profile consists of an administrator and a user. The administrator will typically be the owner of the system and has access to more features than the user. This segregation is based on the assumption, that the administrator has no interest in letting the user control the alarm temperature, since the user is not alarmed by the system. The user is typically the person who has rented the house. The user only has access to a limited number of features.
Both the administrator and the user have access to the system using either the local user interface, or the GSM module. They can both set a specific temperature and turn the heater on or off and inspect the system status. Only the administrator has the possibility of setting the "alarm temperature"[1] to a specific value and receiving alarms from the system.
In order to use the product, the administrator must enter a phone number, that is used to authenticate incoming commands from the GSM module. The phone number also doubles as an access code to the alarm settings.
For the user to use the system, he/she is required to enter a user phone number and the number of days, the profile is to be active. The system will now accept commands from the user as long as the profile is active.

In the description of the use cases no separation between the two means of interfacing with the system, is made. The use cases describe the functionality of the system when using both the GSM module and the local interface. Actions, that are only performed when the GSM module is used is marked with an asterisk (*). The outgoing communication to administrator/user can be

[1] This value indicates the temperature at which the administrator is to be alerted. Consists of both a minimum and a maximum temperature.

established using both the GSM module and the local interface.

3.4.1 Usecase 1 "Set temperature"

Case objective

The administrator or user wishes to change the current set temperature to a specified temperature. In the same action the control of the temperature is commenced. After the case has been executed the system acts as a thermostat, that maintains the set temperature.

Participants

Administrator or user
Temperature sensor
Mock system

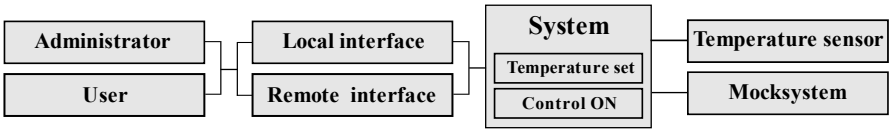

Figure 3.4: Illustration of the participants and functions related to Usecase 1.

Normal system-sequence

1. Administrator or user sets selected temperature.
2. *Used mobile number is compared to authentication list.
3. *Text message syntax is compared to system syntax.
4. Set temperature is compared to legal temperature range.
5. Set temperature is compared to alarm temperature.
6. Confirmation is sent to administrator or user.
7. Temperature sensor measures current temperature of the surroundings.
8. Current temperature is compared to selected temperature.
9. Control unit is activated if current temperature is lower than selected temperature.
10. Action 7 - 9 is repeated in a predefined interval, until the administrator or user turns control off.

Case exceptions

An explanation of the different exceptions can be seen in section 3.4.6 on page 22.

1. Temperature range
2. Alarm temperature conflict
3. *Authentication
4. *Syntax error
5. *Network failure

3.4.2 Use case 2 "Turn temperature control off"

Case objective

The administrator or the user wishes to terminate the control of the temperature. In the same action the system is returned to only monitoring the temperature.

Participants

Mocksystem
Administrator or user

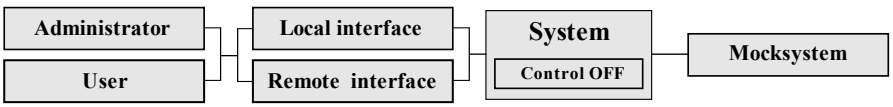

Figure 3.5: Illustration of the participants and functions related to Usecase 2.

Normal system-sequence

1. Administrator or user turns control unit off.
2. *Used mobile number is compared to authentication list.
3. *Text message syntax is compared to system syntax.
4. Confirmation is sent to administrator or user.
5. The control of the temperature is turned off.
6. The system returns to monitoring the temperature (normal operation).

Case exceptions

1. *Authentication
2. *Syntax error

3.4.3 Use case 3 "Status"

Case objective

The administrator or the user wishes to receive information regarding the system status. The system status covers current temperature, alarm temperature and control status.

Participants

Administrator or user
Temperature sensor

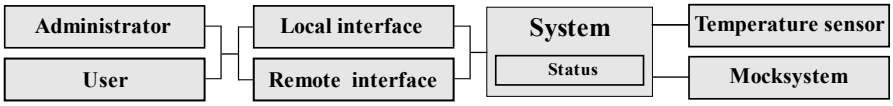

Figure 3.6: Illustration of the participants and functions related to Usecase 3.

Normal system sequence

1. Administrator or user requests system status
2. *Used mobile number is compared to authentication list.
3. *Text message syntax is compared to system syntax.
4. Temperature sensor measures current temperature of the surroundings.
5. System status is sent to administrator or user.

Case exceptions

1. *Authentication
2. *Syntax error
3. *Network failure

3.4.4 Use case 4 "Set alarm temperature"

Case objective

The administrator wishes to change the alarm temperature. The alarm consists of a maximum and a minimum temperature. If the system measures a temperature outside this interval an alarm is issued to the administrator phone number.

Participants

Administrator

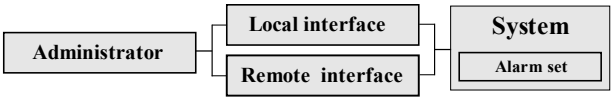

Figure 3.7: Illustration of the participants and functions related to Usecase 4.

Normal system sequence

1. Administrator sets alarm temperature.
2. *Used mobile number is compared to authentication list.
3. *Text message syntax is compared to system syntax.
4. Set temperature is compared to legal temperature range.
5. Confirmation is sent to administrator.
6. System returns to normal operation mode.

Case exceptions

1. Temperature range
2. Set temperature conflict
3. *Authentication
4. *Syntax error
5. *Network failure

3.4.5 Use case 5 "Alarm"

Case objective

During normal operation (only monitoring the temperature) and control operation the system monitors the temperature. If the measured temperature reaches either the minimum or maximum temperature, the system issues an alarm to the administrator.

Participants

Temperature sensor
Administrator

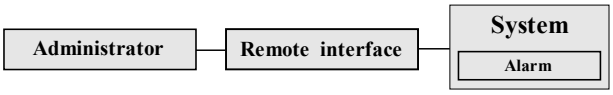

Figure 3.8: Illustration of the participants and functions related to Usecase 5.

Normal system sequence

1. Temperature sensor measures current temperature of the surroundings.
2. The current temperature is compared to the selected alarm temperature.
3. Current temperature is lower than minimum alarm temperature or higher than maximum alarm temperature.
4. System status is sent to administrator.
5. *System status is sent to the administrator every 30 minutes for a maximum of 2 hours, provided that current temperature remains identical to, or higher than selected alarm temperature.
6. Administrator turns alarm off.
7. *Used mobile number is compared to authentication list.
8. *Text message syntax is compared to system syntax.
9. Alarm is turned off.

Case exceptions

1. *Network failure
2. *Authentication
3. *Syntax error

3.4.6 Case exceptions

In this section the case exceptions are described in detail.

Temperature range

This exception is run, when the temperature is not within the specified temperature range[2].

1. Temperature selected by user is out of range according to default settings.
2. Error is sent to administrator or user.

Alarm temperature conflict

This exception is run when the set temperature conflicts with the alarm temperature. This exception will detect, when the temperature is accidentally set so it conflicts with the alarm temperature, simply by comparing the two values.

1. Temperature selected by user, conflicts with alarm temperature.
2. Error is sent to administrator.

Set temperature conflict

This exception is run when the set alarm temperature conflicts with the set temperature. This exception will detect when the temperature is accidentally set so it conflicts with the set temperature, by comparing the two values.

1. Alarm temperature selected by administrator or user, conflicts with the set temperature.
2. Error is sent to administrator or user.

*User authentication

When using the remote interface i.e. the GSM module the used phone number is compared to both the administrator and the user phone number in order to authenticate the number's access to the system.

1. Used mobile number is not registered in authentication list.
2. Error is sent to same number as received text message.

[2]Interval specified in the system specifications 3.5 on the next page

***Syntax error**

When using the remote interface i.e. the GSM module, the received text message, after authentication, is checked according to system syntax. This is done in order to ensure that the system can understand the command.

1. Error is sent to same number as received text message.

***Network failure**

The remote interface is depending on the GSM network, in order to send and receive text messages. This exception is implemented in order to ensure that the administrator or user can always communicate with the system and vice-versa.

1. GSM module can not connect to GSM network.
2. System retries to connect to GSM network.
3. Action 1 - 2 is repeated in a predefined interval.

3.5 Performance specifications

In this section the performance specifications for the system are described. The specifications cover both the functionality driven demands, that are derived directly from the desired product features, and the technology driven demands, that are dictated by the implemented technology. The section also contains a description of the performance specification acceptance tests. The objective of the tests are to evaluate to which degree the product fulfills the performance specifications.

3.5.1 Functionality driven demands

The functionality driven demands are based on the functionality of the system. The demands are derived from the specific tasks, that the system is designed to handle.

I/O devices

Functionality driven demands in relation to the I/O devices e.g. the temperature sensor and the control unit.

- The temperature sensor must be able to measure the temperature within the interval -20 - 100$^{\circ}C$. The range is based on an evaluation of the relevant environment temperature. *The range is obviously not feasible under normal operation in a holiday home.*
- The keypad must be a numeric type keypad with the numbers 0 to 9.
- The LCD must be able to display the menus related to the local user interface.

Communications

Functionality driven demands in relation to the communication between the participants of the system.

- The administrator or user must be able to communicate with the product using a GSM module.
- The product must be able to communicate with a monitoring PC using a RS-232 connection[3] 4.5.
- The GSM module must be able to send and receive text messages.
- The administrator or user must be able to communicate with the system using a local interface.
- The local interface must consist of an LCD and a keypad.
- The LCD must be able to display system information e.g. alarm temperature, current temperature and operating mode.

System

Functionality driven demands in relation to the design of the system.

- The system must update the temperature, time and operating status, at least every 30 seconds.
- The system must be able to determine the temperature using the temperature sensor with an accuracy of $0.5°C$. This accuracy is more than enough to enable the control of the temperature, using the GSM RCS as a thermostat.
- The system must be able to alarm the administrator if the temperature reaches the alarm temperature.
- The set temperature and alarm temperature must be adjustable with $1.0°C$ accuracy.
- The alarm temperature must be adjustable within the range 0-80$°C$
- The set temperature must be adjustable within the range 0-40$°C$
- The system must be able to turn the mock system on and off.
- The system must be able to authenticate the administrator and user access.
- The system must be able to determine if received commands are consistent with system syntax.
- The system must alarm the administrator within 120 seconds after the measurement of an alarm temperature.

[3]The implementation of the RS-232 connection is required to perform the initial programming of the system

3.5.2 Technology driven demands

The technology driven demands are the technology related demands for the product. The demands specify general hardware and software features, that are necessary in order to design an intuitive and usable system.

Communications

Technology driven demands in relation to the communications between the participants and the system.

- The LCD should have a resolution of 64 characters $(16 \cdot 4)$. This allows the design of an intuitive menu setup, with 4 lines for menu points.
- The user interface should be intuitive to use, based on a test where a maximum of 10% of the subjects may find it difficult.
- The text message syntax should be intuitive, based on a test a maximum of 10% of the subjects may find it difficult.
- The incoming SMS commands to the system must have a maximum text length of 15 characters.

I/O devices

Technology driven demands in relation to the I/O devices e.g. the temperature sensor and the control unit.

- The temperature sensor must have a distance to the actual system of at least one meter. The sensor must not be placed in the proximity of the system. This is due to the fact that the operating temperature of the used MPU is well below the desired maximum measurable temperature. In order to secure the functionality of the MPU the temperature sensor and the system is to be placed away from each other.

System

Technology driven demands in relation to the design of the system.

- The system operating frequency should be 8 MHz in accordance to the recommend MPU clock frequency.
- The system software should allow room for additional programming, in relation to implementation of additional features.

3.5.3 Default settings

The default settings allow the administrator or user to use the product without initializing the entire system set up. The default settings among other things state the alarm temperature.

- Alarm temperature is set default to minimum 0^oC and maximum 80^oC.
- The set temperature is set default to 21^oC.

3.6 Performance verification

The conditions for the performance verification tests are described in this section.
The basis for the test is the performance specifications, that dictate the demands that are to be fulfilled. Not all demands are verified, because some tests include several demands. This means that some demands are verified through tests designed to verify other demands.

3.6.1 I/O devices

The system has 5 I/O devices. One of the main I/O devices is the GSM module, but since this is a product delivered by Siemens, it is not to be tested. The next I/O device is the temperature sensor. This is to be tested for accuracy and there is therefore written a test paragraph for this device. The local interface, consisting of the LCD and keypad, is tested for failures. The mock system is also tested.
In order to structure tests for each device, a paragraph is formed for each test:

§ 3.6.1.1 **The temperature sensor must be able to measure the temperature with an accuracy of at least 0.5°C.**
Three thermometers are placed together with the temperature sensor in the vicinity of an adjustable heat source. The heat source is turned on and the temperature is logged from the three thermometers and the temperature sensor. The temperature is raised and the temperature is logged again. This procedure is repeated 9 times.
The measurements are compared and it is determined whether the temperature sensor can measure the temperature with an accuracy of $0.5^{\circ}C$.

§ 3.6.1.2 **The LCD must have 4·16 working characters**
A test program is written in assembly. The program is created to fill the LCD with the same character in all spaces in order to ensure that all characters on the display work. The motivation for this test is that a dead pixel might change the meaning of the information written on the display.

§ 3.6.1.3 **All buttons on the keypad must work**
A test program is written. This program writes the corresponding symbol from the keypad on the display, when a button is pressed. If every button is represented by the correct symbol on the display, the keypad works as intended.

§ 3.6.1.4 **The software has to be able to switch the control unit on and off**
The software has to turn the control unit on and off according to the settings in the software and the measured temperature.

3.6.2 Communications

The following paragraph deals with the communications between hardware related I/O units e.g. the GSM module.

§ 3.6.2.1 **The administrator or user must be able to communicate with the product using a GSM module.**
The system is reset and initialized. Using a mobile phone the status command is sent to the GSM module. If the system replies as expected the demand is considered as fulfilled.

§ 3.6.2.2 **The SMS commands has to give correct response**
The SMS commands in the user manual 3.8 on page 31 is tested one by one and the response is noted. If all responses from the system is as expected, the test is rendered a success.

§ 3.6.2.3 **The local user interface (Menu system) should be intuitive to use, based on a test a maximum of 10% of the subjects may find it difficult.**
A test group consisting of 10 persons is presented the user manual. The test group is to read the manual and state whether they find it difficult or intuitive.

§ 3.6.2.4 **The text message syntax should be intuitive, based on a test a maximum of 10% of the subjects may find it difficult**
A test group consisting of 10 persons is presented the product SMS syntax. The test group is to state whether they find it difficult or intuitive.

3.6.3 System

The following tests are designed to verify the functionality of the system.

§ 3.6.3.1 **The system must update the temperature, time and operating status, at least every 30 seconds.**
The system is reset. The interval with which the system performs an update is measured.

§ 3.6.3.2 **The system must be able to alarm the administrator, if the temperature reaches the selected alarm temperature (maximum and minimum temperature).**
The system is reset. The alarm temperature is set default to 10^oC - 30^oC.
To test the alarm for 30^oC the temperature sensor is placed near a heat source. Using a digital thermometer the time in which the alarm temperature is reached is determined. The time it takes for the system to alarm the user is measured. The same approach is used when testing the alarm for 10^oC, only instead of a heat source, a cooling fan is used.

§ 3.6.3.3 **System endurance test**

The system is set to run over a period of four hours. In this period of time, the system is checked for stability. For every half hour, four commands is sent to the system. The system has to respond correct to the send commands. The remote interface is used for this test. The details for commands and test execution can be found in the system test journal D on page 114

3.7 User manual

The desired functionality of the GSM RCS is illustrated through a user manual, that describes the normal use of the system. This manual is intended to show the features of the GSM RCS and the group's perception of the system as an idea.

Preface

The GSM RCS is a remote control system, designed for remote and local control and monitoring of electrical devices. The embedded GSM module allows the system to be accessed using a GSM mobile phone.
The GSM RCS is not operation-ready in the sense, that it is only possible to control a mock system, that simulates the connection to an actual electrical consumer device. The control and monitoring of actual electrical consumer devices are easily implemented, but are not featured in the GSM RCS at the moment.
The system can be controlled both using SMS text messages and the keypad mounted on the device itself.

3.7.1 Administrator Guide

Getting started

When the system is reset or started up, the program setup is initialized automatically. During the reset/start up, the administrator is required to enter the administrator phone number, PIN code, the time and date. The administrator also has the opportunity to change the operating settings, e.g. the set temperature, alarm temperature and operation mode. These entries are all made, using the numeric keypad.
The first menu displayed on the LCD, during the initialization, is illustrated in table 3.1. In this menu the administrator is asked to enter the administrator phone number. The administrator phone number is the number, that the system will use when sending alarm messages. It will also allow this phone number to remote control/monitor the system. The user phone number does *not* have to be entered during initialization, but it is possible to enter it after the initialization has been completed. The administrator phone number registration menu is illustrated in table 3.1, where the administrator phone number 29661784 has been entered. Notice, that since the system is a prototype and not a commercial product, only mobile numbers from Denmark are accepted by the system.

Administrator Enter Phone number +4529661784

Table 3.1: The menu displayed on the LCD when entering the administrator phone-number.

When the administrator has entered the administrator phone number, he/she is asked to enter the PIN code for the SIM card used in the system. The menu related to this action is illustrated in table 3.2.

Administrator Enter PIN code . . . .

Table 3.2: The menu displayed on the LCD when entering the SIM cards PIN code.

The last part of the initialization, is setting the system time. The time is set to enable the system to determine, when to terminate a user profile. For further information see User options in section 3.7.2. The displayed menu is illustrated in table 3.3.

Administrator Enter Time/date ..:.. ../..

Table 3.3: The menu displayed on the LCD when setting the system time.

When the last step has been completed, the system initializes the GSM module and displays the menu illustrated on figure 3.4. The change from this menu to the administrator main menu indicate, that the system initialization has been completed.

GSM RCS ES Department AAU 2008

Table 3.4: The menu displayed on the LCD during internal initialization

The administrator now has the opportunity to change the default settings and operation mode or

use the system as it is. To avoid any operating problems it is recommended that the administrator check the default settings. This is recommended as these settings might not suit the environment and application in which the system is installed. After the initialization has been completed, the system returns to normal operation.

Local Administrator

When using the local interface, the current system status is displayed on the LCD as shown in 3.5. This menu is displayed on the LCD, when the system is in normal operation. In the status menu the current temperature, time, control unit and GSM status is displayed. To activate the menu system, simply press any key on the keypad.

C. Temp	13^oC
Control	OFF
Time	12.30
GSM	Online

Table 3.5: The menu displayed on the LCD during normal operation.

Main Menu	
User	1
Reset user	2
Administrator	3

Table 3.6: The main menu.

Administrator menu

Select the administrator menu by pressing **3**, while in the main menu. When trying to access the administrator menu, administrator or user must enter the administrator phone number, to confirm administrator rights. The administrator number, is the number, that was entered during the initialization of the system.

Temperature Default: $21°C$
Select Set, enter the desired temperature.
Entering an invalid temperature (out of range) or a temperature conflicting with the set alarm temperatures, will clear the entered values. Please select a different temperature or contact the administrator. The menu can be seen in table 3.13 in the "User Guide".

Alarm Default: $0°C - 80°C$
Select Set, enter the desired temperature.
When one of the set alarm temperature is exceeded, the system will send an SMS stating the current status of the system, including the temperature to the administrator. Notice that setting the alarm temperature, will limit the "Set Temperature" to be within this limit. The Alarm temperature can not be set so the currently set temperature is outside the limit. The user does *not* receive alarm information and is not able to change the alarm settings.

Alarm menu	
Current	0^oC-35^oC
Set	1
Back	2

Table 3.7: The alarm menu displaying the default alarm temperature settings.

Control Select on/off.
Enable or disable temperature control. By disabling the temperature control, the system enters monitoring mode. In this mode the system will only monitor the temperatures and dispatch alarms if alarm limits are exceeded. The menu can be seen in table 3.14, in the "User Guide".

Remote Administrator

Following is a listing of available administrator commands. Some of these are the same as those available to the user. Administrator specific commands, requires the SMS to be sent from the phone number, entered as the administrator phone number.

Action	**Command**	**System SMS response**
Obtain system status.	status	System is online! Current Temperature: xx Temperature control is: On/Off Alarm range is: xx-yy
Activate system and set temperature.	settemp xx*	Temperature Control: Enabled!
Stop thermostat operation and return to monitoring mode.	stop	Temperature Control: Disabled!
Set the alarm temperatures.	setalarm xx yy**	Alarm range has been set to: xx-yy
Turn off alarm function.	alarm off	Alarm disabled.

Table 3.8: SMS syntax table. The * signs is not a part of the syntax.

* 2-digit number representing the temperature in $°C$

** xx is a 2-digit value representing the low alarm temperature and yy is a 2-digit value represents the high alarm temperature.

Error Message	Explanation
Syntax Error!	Syntax error in the SMS send to the system
Alarm Temperature is out of range!	Alarm temperature is set out of the given range
You are unauthorized!	Unauthorized user sends a SMS to the system
Set Temperature is out of range!	The set temperature is out of range
Alarm Temperature conflicts with Set Temperature!	Set Temperature is outside Alarm temperature limits
The set Temperature conflicts with Alarm Temperature!	The set temperature is in conflict with the alarm temperature

Table 3.9: Error Messages

3.7.2 User guide

Local User Options

When using the local interface the current system status is displayed on the LCD, as seen in table 3.5. This menu is displayed on the LCD, when the system is in standby mode. In the status menu the current temperature, time, control unit and GSM status is displayed. To activate the menu system, simply press any key on the keypad.

User menu
Enter the user menu by pressing **1**, while in the main menu.
When using the system for the first time, a user must enter his/her phone number and the number of days the user will be using the system. This is done to enable the user to control the temperature, using the remote interface. The registration menu is illustrated in table 3.10. The days active menu is illustrated in table 3.11. Note the maximum time a profile can be active is limited to 99 days.

User Enter user Phone number +45 . . . . .

Table 3.10: The menu displayed on the LCD when entering the user phone number.

Days active Enter days Active . .

Table 3.11: The menu displayed on the LCD when entering the number of active days.

After the user has been setup, the User Menu is displayed.

User menu	
Temperature	1
Control	2

Table 3.12: The user main menu.

Temperature Select Set, enter the desired temperature.
Entering an invalid temperature (out of range) or a temperature conflicting with the set alarm temperatures, will clear the entered values. Please select a different temperature or contact the administrator.

Temperature	
Current	13^oC
Set	1
Back	2

Table 3.13: The user temperature menu.

Control Enable or disable temperature control.
Select on/off.
Enable or disable temperature control. By disabling the temperature control, the system enters monitoring mode. In this mode the system will only monitor the temperatures and dispatch alarms if alarm limits are exceeded.

Control menu	
Turn on	1
Turn off	2
Back	3

Table 3.14: The user control menu.

Reset user
The user profile can be reset (terminated), by pressing **2**, when in the main menu. The reset user menu is illustrated on figure 3.15

Reset user	
Confirm	1
Back	2

Table 3.15: The reset user menu.

Remote User Options

Following is a list of SMS commands for control of the GSM RCS.

Action	Command	System SMS response
Obtain system status.	status	System is online! Current Temperature: xx Temperature control is: On/Off Alarm range is: xx-yy**
Activate system and set temperature.	settemp xx*	Temperature Control: Enabled!
Stop thermostat operation and return to monitoring mode.	stop	Temperature Control: Disabled!

Table 3.16: SMS syntax table.

* 2-digit number representing the temperature in $^{\circ}C$

** The alarm values are Administrator controlled, if you encounter problems due to these, please contact the administrator.

Error Message	Explanation
Syntax Error!	Syntax error in the SMS send to the system
You are unauthorized!	Unauthorized user, or user trying to send administrator command to the system
Set Temperature is out of range!	The set temperature is out of range
The set Temperature conflicts with Alarm Temperature!	The set temperature is in conflict with the alarm temperature (administrator set)

Table 3.17: Error Messages.

Chapter 4

Hardware Design

This chapter describes the design phase of the hardware, which consists of different key modules. For this semester it is required that each group creates a running minimal system. Therefore the first modules described in this chapter, are those of the minimal system. Afterward the rest of the hardware needed, to create the remote control system, is described and added to the minimal system. The needed hardware is in accordance with the demands in section 3.5.
A block diagram has been created to visualize the different blocks, of both the complete system and the minimal system. This block diagram is shown in figure 4.1. This makes it easy to see which modules are included in the minimal system and which are specific for the functionality of this system.

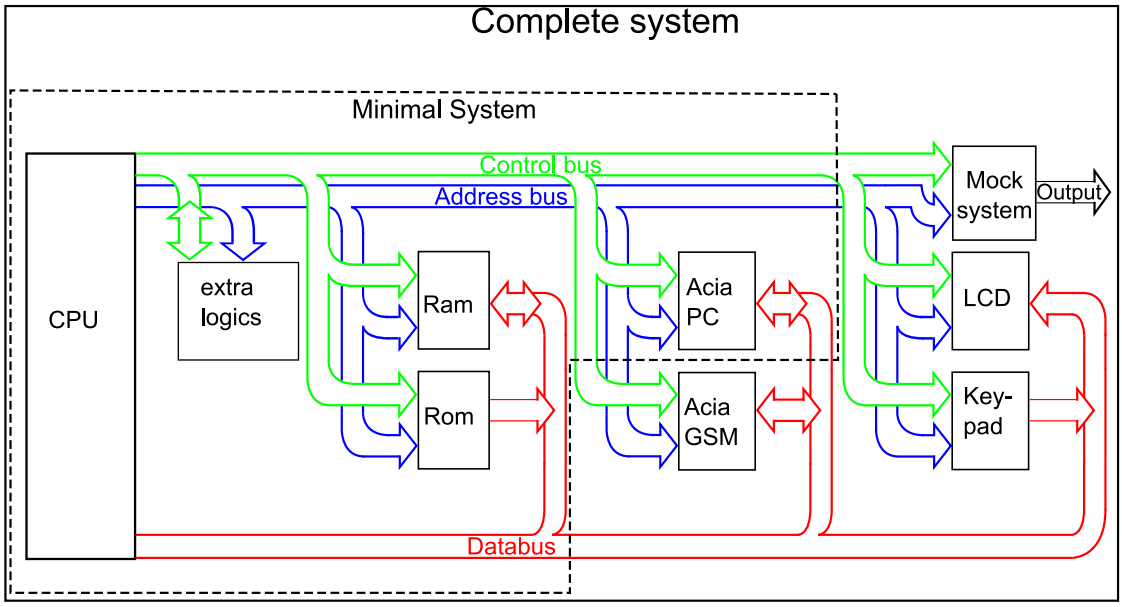

Figure 4.1: Block diagram of the minimal system

4.1 The minimal system

The main blocks in the minimal system are shown in figure 4.1. The minimal system is marked and the components are described in this section.

4.1.1 CPU

The CPU or processor is the unit which accepts and executes commands from the memory. The system is based on a Motorola 68000 (M68k), which is a 32 bit processor. The processor is compatible with synchronous and asynchronous data transfer, 4.11 on page 62. The processor handles bytes(8 bits), words(16 bits) and longwords (32 bits). The layout for the processor is shown in figure 4.2.

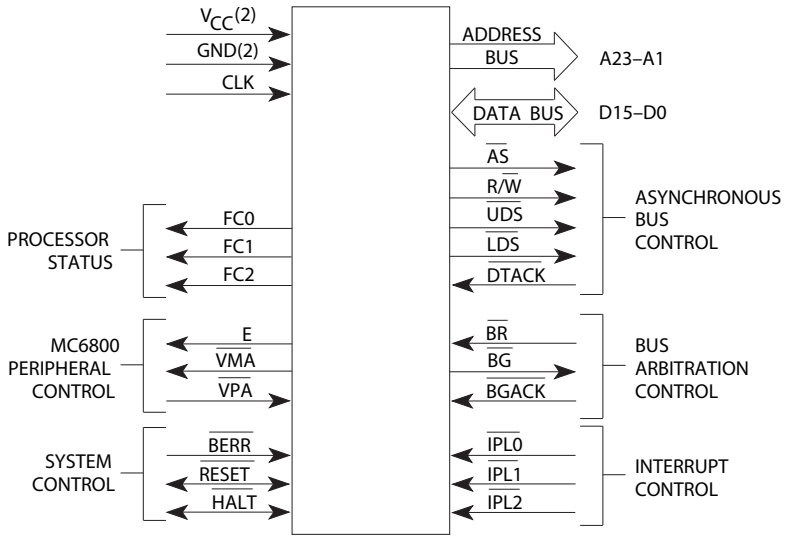

Figure 4.2: Motorola 68000 in/out-put diagram

Essential pins

The reason for the $V_{cc}^{(2)}$ and $GND^{(2)}$, is that there is two of each. These pins are located on each side of the processor and their function is to avoid voltage drops inside the processor. The CLK input is for the clock, which determines the speed at which the processor operates. The processor is normally recommended to run at 8 MHz, so therefore the clock speed is selected to 8 MHz. This should be more than sufficient for the functionality of this system.

Processor status pins

The three outputs $FC_0 - FC_2$, indicate the processor status, as shown in figure 4.2. These pins are only used to indicate address space in this system. This is indicated when the processor asserts the three pins together. This means that the processor is in an interrupt routine. For more information on the processor status pins, see the data sheet [9, table 3-3, p. 3-9].

Peripheral control pins

The M68k has three pins for peripheral control, E, $\overline{VMA}$ and $\overline{VPA}$. E stands for Enable and is an output from the processor. E can be used to activate peripheral circuits. A single clock cycle of E has the length of 10 M68k clock cycles, E is low for six and high for four M68k clock cycles.
$\overline{VMA}$ stands for Valid Memory Address and is an output from the processor. This pin indicates to the peripheral devices, that the address on the address bus is valid and the processor is synchronized with the E signal.

$\overline{VPA}$ stands for Valid Peripheral Address. This input indicates that the addressed device or memory is a M6800 familiar device and data transfer should be synchronized with the E signal.

System control pins

The processor has three system control pins, $\overline{BERR}$, $\overline{RESET}$ and $\overline{HALT}$.
$\overline{BERR}$ stands for bus error. This is a input pin to indicate a bus error in the current bus cycle to the processor. There is no bus controller in this system and therefore this input is not used. $\overline{BERR}$ is therefore connected directly to the supply to prevent this signal from being active at anytime.
$\overline{RESET}$ is an input/output. If reset is asserted externally, it resets the processor. As an output, the processor can reset other devices. If reset is asserted alone, all external devices are reset without affecting the internal state of the processor. $\overline{RESET}$ and $\overline{HALT}$ must be asserted at the same time, to create a complete system reset.
$\overline{HALT}$ is an input, and an active signal, at this pin, will cause the processor to stop bus activity at the end of the current bus cycle. In the GSM RCS, $\overline{HALT}$ and $\overline{RESET}$ are only used to create a complete system reset and are therefore connected to the same reset signal to ensure that they are asserted at the same time.

Address bus pins

The address bus has the pins A_1-A_{23} and is therefore a 23 bit address bus. There is no A_0 on this processors address bus, instead the processor has two pins, Lower Data Strobe, LDS, and Upper Data Strobe, UDS. These two pins will be described in relation with the asynchronous bus control, 4.11.2 on page 63. The address bus is capable of addressing 16 Mbytes of data.

Data bus pins

The processors 16 bit data bus has the pins D_0-D_{15}. The data bus can both accept and send data of word or byte length. During an interrupt acknowledge cycle, the external device supplies the vector number on data pins D_0-D_7.

Asynchronous bus control pins

The Asynchronous bus control consists of $\overline{AS}$ (Address strobe), R/$\overline{W}$ (Read/$\overline{Write}$), $\overline{DTACK}$ (Data transfer acknowledge), $\overline{LDS}$ and $\overline{UDS}$.
$\overline{AS}$ is an output, that confirms that a valid address is available on the address bus.
The R/$\overline{W}$ output defines, whether the transfer cycle on the data bus is read or write.
$\overline{DTACK}$ is an input, used to tell the processor when a data transfer is complete. When the processor registers $\overline{DTACK}$ during a read cycle, the data is latched and the cycle is terminated. If $\overline{DTACK}$ is registered during a write cycle, the cycle is terminated.
$\overline{LDS}$ and $\overline{UDS}$ are outputs and control the flow of data on the data bus.
The system memory is arranged in 8 bit banks. These banks are connected to the data bus, so one bank has the lower 8 pins on the data bus and the other bank has the upper 8 pins. $\overline{LDS}$ and $\overline{UDS}$ selects the valid data bits on the data bus. More information on the data strobe control and the data bus, can be found in the data sheet for the processor [9, p. 3-4 & 3-5].

Bus arbitration control pins

The bus arbitration control consists of three pins, $\overline{BR}$ (bus request), $\overline{BG}$ (bus grant) and $\overline{BGACK}$ (bus grant acknowledge).
$\overline{BR}$ is an input which tells the processor, that another device needs to become bus master. $\overline{BG}$ is an output which indicates to other devices, that the processor will release bus control at the end of the current cycle. $\overline{BGACK}$ is an input, that indicate to the processor that another device has become bus master.
The bus arbitration control is not used in this system and these three pins are connected to the supply, in order to prevent them from becoming active at any time.

Interrupt control pins

The interrupt control pins consist of $\overline{IPL0}$-$\overline{IPL2}$. These pins indicate devices requesting interrupts. Level 7 is the highest priority and cannot be masked. Level zero indicates, that no interrupts are requested. The interrupt signals must be asserted, until the processor signals interrupt acknowledge, by asserting $\overline{FC0}$-$\overline{FC2}$ and $A_1 - A_3$.

4.1.2 Clock generator

In order to determine what component to use as a clock signal generator, the demands from the processor must be taken into consideration. The easiest way to get a clock signal, would be from a crystal oscillator. Therefore the M68k demands for clock input are compared to data for crystal oscillators. The clock signal must be at a constant frequency, with a square wave signal. The electrical characteristics for the demands and the crystal oscillators are compared in table 4.1.

Parameter	**Demands for M68k**	**MCO 1425 B**
Pulse width	55 - 125 ns	50 - 75 ns
Rise/fall-time	<10 ns	<15 ns
Low signal voltage	<0.8 V	<0.4 V
High signal voltage	>2 V	>2.4 V

Table 4.1: Clock comparison

The rise/fall time is too slow for the processor, but because the oscillator has a margin of minimum 0.4 V compared to the processor for both low and high signals. The rise/fall time, the processor registers on the clock input pin, is lower than the value stated in the data sheet for the oscillator. Therefore the minimum shiftrate for the processor and the oscillator are calculated to determine whether, the oscillator is useable or not.

$$shiftrate_{osc} = \frac{2.4 - 0.4V}{15ns} \tag{4.1}$$

$$= 133.33 \cdot 10^6 V/s \tag{4.2}$$

Then the minimum shiftrate for the processor is calculated:

$$shiftrate_{cpu} = \frac{2 - 0.8V}{10ns} \tag{4.3}$$

$$= 120 \cdot 10^6 V/s \tag{4.4}$$

By comparing these two simple calculations it becomes obvious, that the oscillator is sufficiently fast and is therefore selected as clock oscillator for the processor.

4.1.3 Power-on-reset circuit

The processor must have a power-on-reset circuit in order to function. This circuits function is to reset the processor when the system is turned on. This circuit can also be used to reset the system, while running if necessary. In order to reset while running, it is necessary to implement a button to activate this function.

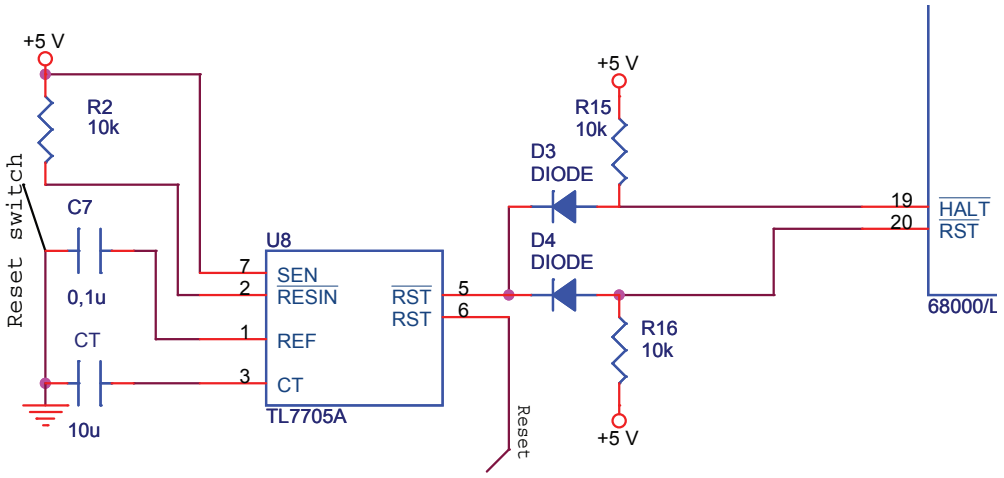

Figure 4.3: Power-on-reset circuit

It is stated earlier in this section that the processor has two pins, $\overline{RESET}$(pin 20) and $\overline{HALT}$(pin 19), which must be asserted together for at least 100 ms to reset the processor, See figure 4.3. Normally the implementation of a reset button, requires a bounce filter, but the TL7705A circuit starts to hold the reset state, when the button is first initiated. TL7705A will not react to changes in state from the button, until 100 ms later.
A TL7705 component is built for reset circuits for microprocessors and the datasheet states, that a capacitor is in control of the reset time of the circuit. The following formula is given:

$$t_d = 1.3 \cdot 10^4 \cdot C_T \tag{4.5}$$

Since the reset time is known, the size of the capacitor is the only missing parameter. This parameter can be extracted from equation 4.5 and the value for the capacitor C_T is calculated to:

$$C_T = \frac{t_d}{1.3 \cdot 10^4} \tag{4.6}$$

$$= \frac{100ms}{1.3 \cdot 10^4} \tag{4.7}$$

$$= 7.69\mu F \tag{4.8}$$

Unfortunately, there is no capacitor with this value, so therefore the used capacitor will be the nearest larger model. A 10 μF capacitor is used to control the reset time.

4.1.4 Memory

The system has two types of memory, ROM (Read Only Memory) and RAM (Random Access Memory). The ROM contains the program code. The processor loads the program from the ROM and then uses the RAM as temporary storage space for the different variables in the software. The ROM used in this system is AMD's Am29F010 with a capacity of 131.072 x 8 bits. The used RAM, K6T4008C1C, is made by Samsung and has a capacity of 512.000 x 8 bits. The pin connections for the RAM and ROM are shown in diagram 4.4. The RAM and the ROM have most of the pins in common. The RAM has two pins more for

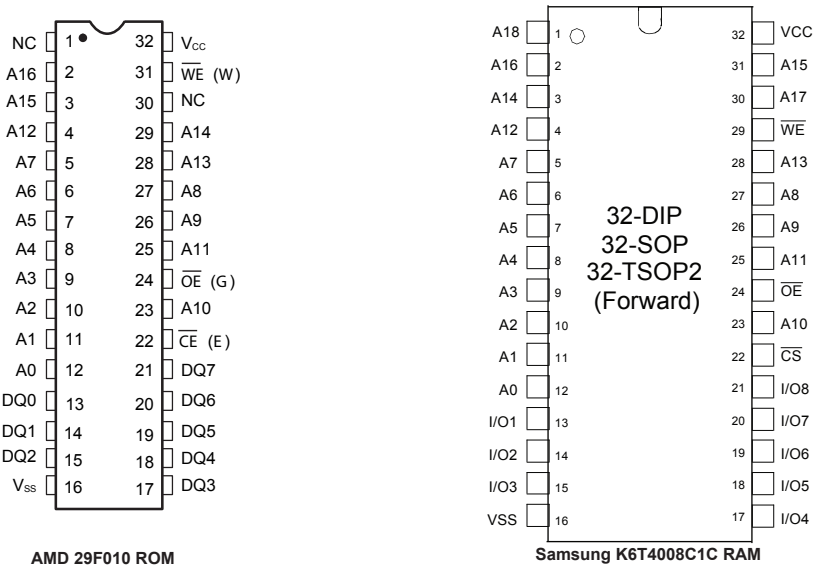

Figure 4.4: Pin connections for the memory banks

the address bus, than the ROM. The extra pins are required to address the 4 times larger memory in each block. The address bus is labeled A0-A16 for the ROM and A0-A18 for the RAM, see figure 4.4. Both RAM and ROM have 8 bit data buses. On the ROM the data bus is denoted DQ0-DQ7 and on the RAM, I/O1-I/O8. These buses are connected respectively to the address bus and the data bus on the processor.
However since the processor has a 16 bit data bus the memory banks, have to be arranged in parallel, to gain a 16 bit bandwidth on the data bus. This is obtained by connecting one bank to the lowest 8 pins on the processors data bus (D0-D7) and the other bank to the rest (D8-D15). This setup is used for both RAM and ROM and therefore the minimal system consists of two of each.
Both RAM and ROM have some control pins to control the flow of data onto, or from the memory. These control pins consist of a $\overline{CS}$ (chip select) input, $\overline{OE}$ (Output enable) and $\overline{WE}$ (Write enable). The program upon completion, is to be burned into the ROM and therefore there will be no reason to enable the system to activate the $\overline{WE}$ on the ROM. $\overline{WE}$ is therefore connected to the supply. $\overline{OE}$ is connected to $\overline{OE}$ on peel 1, which is generated from the R/$\overline{W}$ pin on the processor, see section 4.3 on page 43. The RAM is connected in an similar manner, but instead of connecting the $\overline{WE}$ to the supply, this is connected to the $\overline{WE}$ pin on peel 1, which is also generated from the R/$\overline{W}$ pin on the processor. Again $\overline{OE}$ is connected to $\overline{OE}$ on peel 1. For more information on the RAM, check the data sheet [11, p. 3].

4.2 Memory and Addressing

In the minimal system, the debugging/monitoring program TS2monitor is used. The program has a set of requirements, that must be fulfilled, for the program to work. This means that some of the components in the system, will be required to respond on certain addresses. The addressing requirements are as following:

- The TS2 monitor must be allocated to the ROM at address 0x0000 to 0x1200
- The program must have RAM at addresses 0x40000 to 0x40FFF
- Serial PC communication on addresses 0x800001 to 0x800003

Additional requirements:

- Additional RAM and ROM is needed to store and execute the program, which the TS2mon is used to debug.

4.2.1 Memory types

The name, Read Only Memory, indicates that the data in the chip, is permanent and unchangeable through software. However, this is not entirely true, as the chip can be reprogrammed through a flash-procedure. This procedure is very time consuming and the chip is only guaranteed to about 100.000 writes/rewrites. Therefore, the ROM will be treated as Read Only.
Random Access Memory, is a temporary memory, which can be read and written to much faster than the ROM. The downside of the RAM is, that the content will be lost, when power is turned off. This is why the final program will be stored in the ROM, but all temporary data, only needed when the system is operating, will be stored in the RAM.

4.2.2 Addressing

The M68k processor supports addressing in the range from 0x000000 to 0xFFFFFF, which leaves plenty of addresses for extending the ROM, RAM and peripherals in the final system. The mapping required for the TS2monitor, leaves just enough addresses to address another ROM module continuous in the address space. The arrangement can be seen in the memory map in table 4.2. The processor also operates with 23 address pins, ranging from A_1 to A_{23}, where most processors would have 24 pins, starting from A_0. The data bus on the M68k processor operates in 16 bit, but can use the data bus in both 8 bit and 16 bit mode, and can access both the upper and the lower part of the bus, independently. Accessing all 16 bits at once will require separate memory modules to be connected to the upper and lower part of the bus.
Since the ROM and RAM modules only have an 8 bit bus, it is necessary to connect them in parallel to each other, allowing the processor to use both 8 bit and 16 bit mode on the data bus, since the instruction set in some cases requires all 16 bit. This means that one of each RAM- and ROM-modules must be connected to the upper part of the data bus, and the other to the lower part.

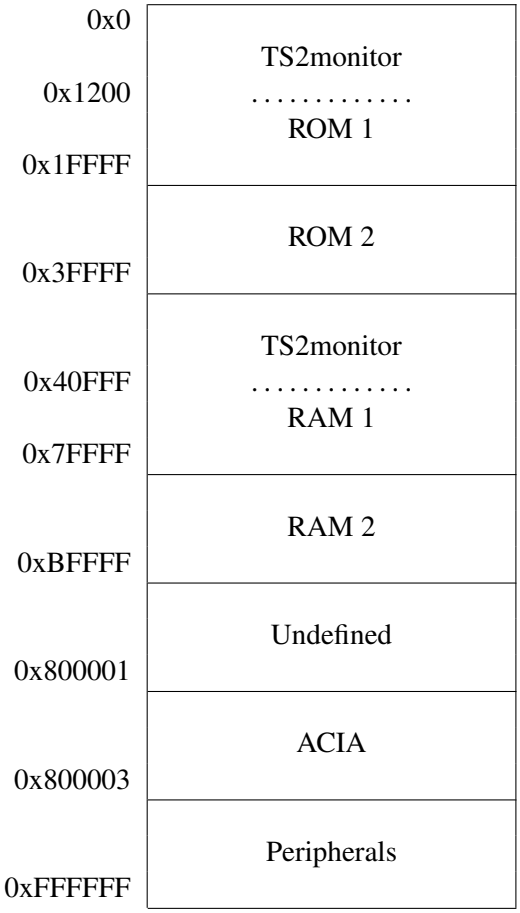

Table 4.2: Memory map for the minimum system.

4.2.3 Chip select

To prevent several modules from reading or writing on the data bus at the same time, it is necessary to use a chip select strobe. The chip select input on the different modules, must be asserted for the module to respond to any address strobes and read/write commands. To create this chip select strobe, an address decoder is used. This enables the programmer to control which system modules are selected.

From truth table 4.3, it can be seen that the maximum bits needed to fully address the RAM, ROM and several peripherals, are the upper 6 bits of the address bus. The pins A_{18} and A_{19} will be the primary pins to differentiate between the RAM and ROM. The ROM needs both pin A_{18} and A_{19} to be logic 0, while the RAM must be selected for A_{18} and/or A_{19} being logic 1.
All peripherals can be chip selected using pin A_{20} to A_{23}.

Module	A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17-2}	A_1	LDS	UDS	$\overline{AS}$
ROM1	0	0	0	0	0	0	X	X	0	1	0
ROM2	0	0	0	0	0	0	X	X	1	0	0
RAM1	0	0	0	0	X	X	X	X	0	1	0
RAM2	0	0	0	0	X	X	X	X	1	0	0
UART	1	0	0	0	0	0	0	X	1	0	0

Table 4.3: Truth table for addressing of ROM, RAM and the serial connection. Values 1 or 0 is the required state on the address bus, X's are "don't cares"

The inclusion of $\overline{AS}$ is needed, to avert false or random chip selects. $\overline{AS}$, Address strobe is a signal the processor sends, when an address is ready on the address bus. By implementing $\overline{AS}$ in the chip select logic, it is possible to avoid any chip select from being generated, before the address is ready on the bus.

4.3 PEEL

To handle address decoding a PEEL chip is used, minimizing the use of actual logic gates. The PEEL is a programmable chip, which can be programmed to act as several connections of different gates. This means, that all the logic used for the address decoding can be done using PEEL chips. The programming for the PEEL is done by using the programming tool ABEL. The ABEL code is easily translated from boolean equations in table 4.4. By calling the different addresses on the M68k, the appropriate chip select signals will be generated by the PEEL's. This ensures that only the intended chips or peripherals react to the call.

ROM1:	$A_{18}+A_{19}+A_{20}+A_{21}+A_{22}+A_{23}+UDS$
ROM2:	$\overline{A_{18}+A_{19}+A_{20}+A_{21}+A_{22}+A_{23}}+LDS$
RAM1:	$A_{23}+A_{22}+A_{21}+(A_{20}*\overline{(A_{19}+A_{18})})+(\overline{A_{20}}*(A_{19}+A_{18}))+UDS$
RAM2:	$A_{23}+A_{22}+A_{21}+(A_{20}*\overline{(A_{19}+A_{18})})+(\overline{A_{20}}*(A_{19}+A_{18}))+LDS$
UART:	$\overline{A_{23}}+A_{20}+A_{21}+A_{22}+VMA$

Table 4.4: Logic equations for generating Chipselect signals.

In the table, only chip selects are listed, but the PEEL's also handle output enable, write enable and data acknowledge signals. The PEEL codes can be seen in Appendix G.

4.4 GSM module

In order to enable the user to communicate with the system using the GSM network, a GSM module is implemented in the system. The used module is a prefabricated Siemens TC-35 GSM module [16]. Using a prefabricated module eliminates a large part of the hardware design. The prefabricated module is designed so it can be used through a RS-232 serial connection.

4.4.1 Location

The GSM module will be connected to a RS-232 that is designed for this purpose. The RS-232 connection is described in section 4.5 on the next page.

4.4.2 Functionality description

As pointed out previously, a prefabricated GSM module will be used. For more details about the circuit see the datasheet [16]. Instead of designing an actual circuit, this section describes all the system requirements, that are needed in order to use the GSM module.
The power supply for the GSM module is provided trough a 6 pin connection. The pin configuration is shown in table 4.5.

Pin	signal	Value
1	Power supply	+Vcc(8-30V)
2	Not used	NULL
3	Power down	$\geq 5V$
4	Ignition	$\geq 5V$
5	Not used	NULL
6	GND	GND(0V)

Table 4.5: Power connection pin setup for GSM module

Apart from the voltage level, the power-down and ignition signal are specified to be asserted for a time before they are registered. The power-down pin has to be active for at least $3.5s$ and the ignition for $200ms$. The ignition function, however, can also be activated from the RS-232 connection. The module uses an external power supply and differs from the rest of the system as it uses 8-30V and not 5.0V. This issue will not be assessed further, as the main focus of the project is not the means of supplying the system.

4.4.3 Circuit design

The module uses the RS-232 connection, that is identical to the connection used by the TS2 monitor. The connection for the GSM module has 9 pins, but not all 9 pins are needed in order to establish communication, as described in section 4.5. The connection is established using asynchronous communication, where software controls the data flow. This configuration only needs to use pin 2 and pin 3 to control the module. Furthermore, the GSM module needs to be activated in order to exchange data. This is done by connecting pin 4, the DTR (Data Terminal Ready) pin, to the reset signal from the M68k. This will cause the GSM module to reset and start up whenever the system is activated.

4.4.4 Construction

The GSM module is a prefabricated module, and the diagrams and specifications can be found in the technical documentation[16]. The information relating to the RS-232 interface can be found in section 4.5.

4.4.5 Conclusion

When using a prefabricated GSM module, the hardware design in relation to the GSM module is limited to the RS-232 interface. Therefore the only actual hardware design consists of connection between the minimal system and the GSM module.

4.5 RS-232 Connection

The RS-232 connections are required by the minimum system for communication with the TS2-monitor and the GSM module.

4.5.1 Location

The RS-232 connection blocks, are placed as parallel peripherals for the M68k. On figure 4.1 on page 35 the location of the RS-232 connections blocks in relation to the M68k, can be seen.

4.5.2 Functionality description

The RS-232 connection communicates by sending all data bits in a row, as illustrated in figure 4.5.

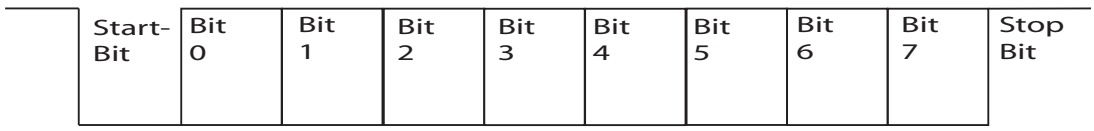

Figure 4.5: Serial bit arrangement in relation to RS-232 communication.

As shown in the figure, the signal begins with a start bit and ends with a stop bit. These signals are not a single bit but series of bits. A series of high bits, indicate the beginning and a series of low bits, indicate the ending. This is the result of using software flow control. This only requires 2 active pins to communicate on the RS-232 connections. Setting up the M68k to use serial communication, the ACIA circuits is handled by the PEEL. In this circuit design it is assumed, that the M68k is correctly setup, when the RS-232 blocks are activated.

4.5.3 Circuit design

To make serial communication possible for the system, an ACIA compatible with the M68k is used. The ACIA converts 8 bit parallel data into serial data rows. The ACIA used in this project is the model 6850[8]. The 6850 has a 8 bit data bus, along with the control pins seen in table 4.6.

Pin Name	Function
E	Enable
R/W	Read/Write
TxCLK	Transmit Clock
CS0,CS1,CS2	Chip select
Reg Sel	Register select

Table 4.6: ACIA control pins.

The enable pin (E) is connected directly to the E pin on the CPU. The $\overline{CS}$ pins are for the addressing, which are controlled by the PEEL circuits, as described in section 4.3. Along with the register select pin, the Read/Write pin controls which register to work with, as shown in table 4.7.

Reg Sel	R/W	Register
1	0	Tx Data Register
1	1	Rx Data Register
0	0	Control Register
0	1	Status Register

Table 4.7: ACIA Registers.

Configurating the ACIA is done by writing to the control register. When this option is selected, the data bus is used to setup the ACIA. Each pin has its own function, when configurating the ACIA. These functions are shown in figure 4.6.

Pin 0	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7
Counter divide		Format select			Transmit control		RxINT Enable

Figure 4.6: Control register setup.

The counter is used to control the bit rate. This is examined in section 4.5.3. The Format select, controls the syntax of the serial data stream. In this project the format for the ACIA used to the GSM module, is set to 8 bit data, no parity and 1 stop bit. This is chosen as it is the default setting for the GSM module. The transmit control and RxINT determines, if the interrupt for sending and receiving data is active. This is setup so it fits the actual system situation. The setup for the ACIA, used for the TS2 monitor is setup automatically by TS2 software program. This is set to: 8 bit data, no parity and 1 stop bit.
The status register is read only. Here it is possible to get the current status of the ACIA, as shown in figure 4.7.

Pin 0	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7
RDRF	TDRE	DCD	CTS	FE	OVRN	PE	IRQ

Figure 4.7: Status register.

Each pin gives the status for a specific parameter, the different flags are shown in table 4.8.

IRQ	Interrupt request flag
PE	Parity error
OVRN	Overrun flag
FE	Frame error flag
CTS	Clear to send flag
DCD	Data carrier detect flag
TDRE	Transmit data register empty
RDRF	Receive data register full

Table 4.8: Pin specifications.

The voltage output from the ACIA is $0-5V$, where the RS-232 standard specifies a ± 3 V voltage level to indicate high and low signals. This is implemented by using a MAX232 [7]. The MAX232 is a line driver/receiver, that by means of 4 capacitors converts the logical voltages levels from the ACIA to ± 10 V, as are usable the RS-232 connection.

Baud rate

In this section two RS-232 connections are made. The only difference between these are the transfer rate. The GSM module runs with a bit rate of 19200bps, where the TS-2 monitor uses 9600bps. In this project an oscillator is used to generate a 4.915.200Hz signal.

$$\frac{Frequency}{ACIAdivider \cdot counterdivider} = Baudrate \tag{4.9}$$

$$\frac{4.915.200}{16 \cdot 16} = 19200Baud \tag{4.10}$$

$$\frac{4.915.200}{16 \cdot 32} = 9600Baud \tag{4.11}$$

This signal is divided partly by means of a counter and partly by the internal function of the ACIA. The ACIA is set to divide the frequency by 16 and the counter divides by 16 for the GSM-connection and by 32 for the TS-2 connection.

Circuit schematic

The finished circuit design is shown in figure 4.8, where all connections and components are placed in the right order. The integration of this part of the system will be described in section 4.12.

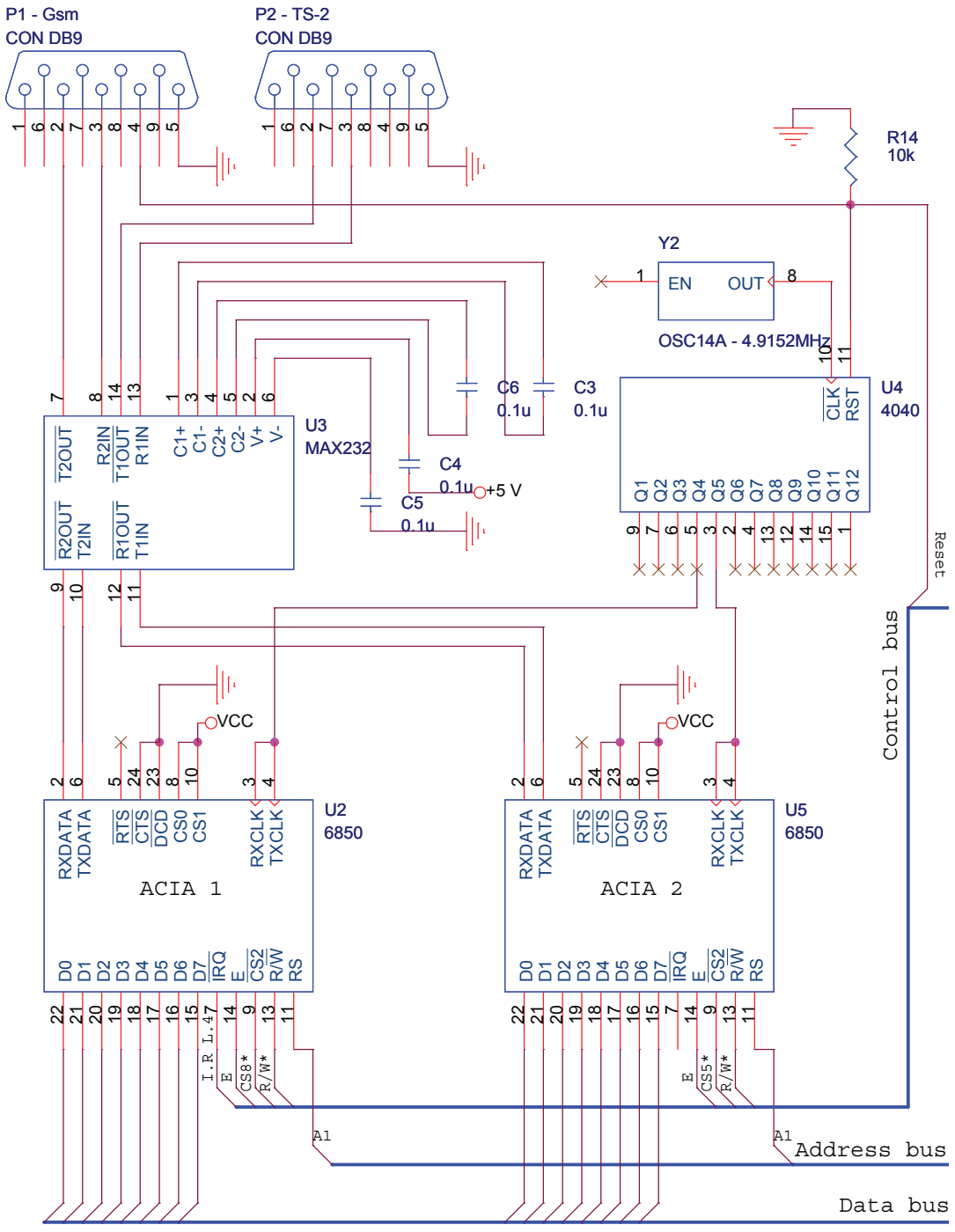

Figure 4.8: Circuit of the RS-232 connections.

4.5.4 Conclusion

The ACIA and the MAX232 circuits are used to construct the RS-232 connection blocks. By moving all logic to the PEEL circuits, the RS-232 connections are relatively simple blocks to design and construct.

4.6 Temperature Sensor and ADC

The main functionality of the system lies in its ability to measure the temperature in the location, where the product is located. In this project, this functionality is enabled by using a temperature sensor. The used sensor is a LM335 [12] precision temperature sensor, which is basically a zener diode. The diode has a breakdown voltage which is directly proportional to the temperature. The LM335 [12] is an analogue device and the minimal system is purely digital. In order for the two terminologies to interact, the analog signal from the sensor must be converted to a digital signal. This is achieved by using the ADC (analog to digital converter) PM0820FP [14].

4.6.1 Location

The LM335 will be connected to the M68k, through the ADC, which will be connected to the data bus. Because the LM335 is in a TO-92 package, it has the advantage of being very inexpensive and takes up very little space.
LM335 consists of three pins, one cathode, one anode and one to calibrate the temperature sensor with. The cathode is connected to ground, and the anode is on V_{in} of the PM0820FP. The calibration pin is connected to a adjustable voltage divider, which is placed between the anode and cathode. When using the PM0820FP, it is configured as a standard 8-bit converter [14, p. 17]. DB0-DB7 pins are connected to the data bus and $\overline{CS}$, $\overline{RD}$ and $\overline{INT}$, are connected to the control bus.

4.6.2 Functionality description

In accordance with the specification demands, the sensor has to cover a temperature area of $-20°C$ to $100°C$ with a resolution of 0.5^oC. This configuration results in $120/0.5 = 240$ steps. This will require an ADC that can handle at least 240 steps. By using an 8 bit ADC, the wanted resolution can be accomplished. An 8 bit ADC can convert a signal into an 8 bit value, and has $2^8 = 256$ steps. An 8 bit configuration allows the use of the, in relation with the M68k, recommended ADC PM0820FP[14].
The PM0820FP[14] has a resolution of 8 bits and has a very low conversion time, which is ideal, since it must be able to detect differences fast. It also runs on a single supply and can operate at 5.0 V.
The LM335[12] is linear, which means that the output voltage from the sensor can be expressed by the function 4.12. This function is also given in the datasheet for the LM335 [12].

$$V_{out_T}(T) = V_{out_{T_0}} \cdot \frac{T}{T_0} \tag{4.12}$$

$V_{out_{T_0}}$ is the output voltage at $25°C$, T is the temperature and T_0 is the reference temperature, $25°C$. Temperatures must be given i Kelvin. The reference voltage, $V_{out_{T_0}}$, is given as $2.98V$. From this expression, the output voltage can be rewritten as a function of the temperature, see eq. 4.14, where T is given temperature in degrees Celsius.

$$V_{out_T}(T) = V_{out_{T_0}} \cdot \frac{T}{T_0} \tag{4.13}$$

$$V_{out_T}(T) = 2.98V \cdot \frac{T + 273.15K}{298.15K} \tag{4.14}$$

This can be illustrated graphically as shown in figure 4.9:

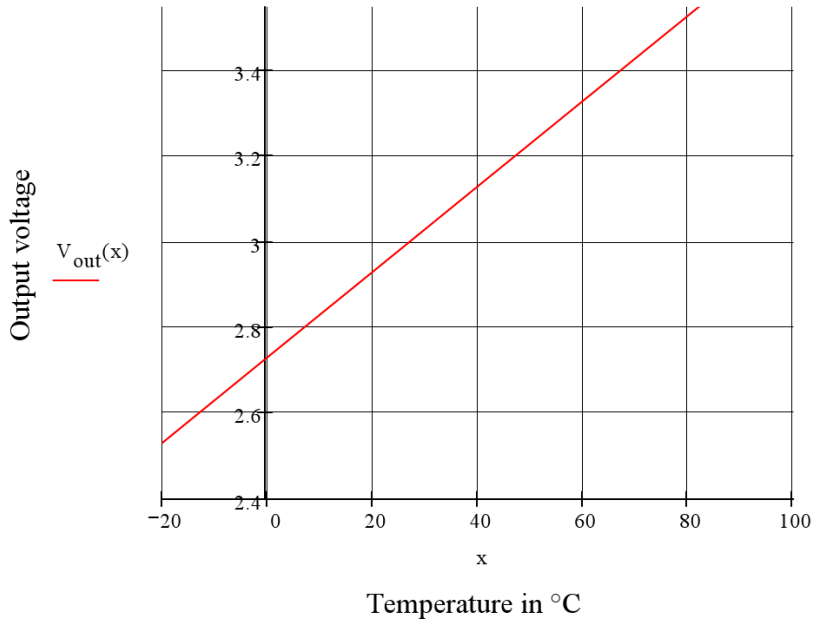

Figure 4.9: Output voltage from LM335 as a function of the temperature, T, in degrees Kelvin.

The sensor will operate from $-20°C$ to $100°C$. The output voltage V_{out_T} can be calculated using eq. 4.14.

$$V_{out_T,min}(T) = 2.98V \cdot \frac{-20+273.15}{298.15} = 2.53V \tag{4.15}$$

$$V_{out_T,max}(T) = 2.98V \cdot \frac{100+273.15}{298.15} = 3.73V \tag{4.16}$$

Concluding that the output voltage will span from $2.53V$ to $3.73V$. Furthermore the temperature area, $-20°C$ to $100°C$, results in a voltage difference of:

$$\Delta V_{out_T} = 3.73 - 2.53 = 1.2V \tag{4.17}$$

By dividing the temperature area with the voltage difference, the voltage per celcius is obtained.

$$\frac{\Delta V_{out_T}}{\Delta T} \Rightarrow \frac{1.2}{120}\frac{V}{°K} = 0.01\frac{V}{°C} \tag{4.18}$$

This means that for every degree kelvin the temperature rises, the output voltage of the LM335[12] will rise 0.01 volts. Also by using the statement in 4.17, the output voltage in terms of bits, can be calculated.

$$1LSB = \frac{\Delta V_{out_T}}{256bits} = \frac{1.2}{256}\frac{V}{bit} = 0.004688\frac{V}{bit} \tag{4.19}$$

This means, that every time the ADC counts 1 bit up or down, there is a change of $4.688mV$. Therefore the ADC must have $4.688mV$ as 1 LSB. To do this the ADC needs $3.73V$ and $2.53V$ as its boundaries. This is done by using the V_{ref+} and V_{ref-} ports on the PM0820FP[14]. By setting V_{ref+} and V_{ref-} to $3.73V$ and $2.53V$, respectively, the ADC will convert the maximum output voltage from the LM335[12] to binary 11111111_2, given that the temperature will not exceed 100°C. It is the same case for a temperature of $-20°C$, which will result in 00000000_2, binary. If the voltage exceeds the given reference voltages, the ADC will give its maximum or minimum value, depending of which reference is exceeded.

4.6.3 Circuit design

In figure 4.10, the temperature sensor can be seen. When pin 7 on the ADC[14] is held low, it will go into Read mode. This means a complete conversion is done by pulling $\overline{RD}$ until, output data appears. D0-D7 is connected to the data bus, which is where the converted analog signal is located.
The temperature sensor requires a constant current to flow, so a LM334[13] is used. V_+ is connected to 5 V, and V_- as well as R is connected to V_{in} on the ADC[14]. R, however, is connected through a 67Ω resistor. The LM335[12] has a $+$, $-$, and an ADJ pin. The $+$ is the "output" of the temperature sensor and is connected to V_{in}. $-$ is connected to GND. ADJ is set to a variable resistor. ADJ is used to calibrate the temperature sensor so that at $25°C$, the LM335[12] has an output of 2.98V
To get the desired digital values, a reference voltage has to be implemented, so the analog signal is converted to the correct binary value. To do this, the reference voltages are set to 2.53V and 3.72V, respectively. The ADC[14] has two pins for the reference voltages. Ref_+ and Ref_-. Each pin has their own, but similar, circuit to get the desired voltage. On Ref_+ a voltage divider between V_{CC} and GND is used, to send 3.72V on to the pin. The 3.72V is sent through a buffer to ensure that the Ref_+ pin can pull as much current as needed, without affecting the voltage. The same is the case for Ref_-, where a voltage divider ensures a voltage of 2.53V.

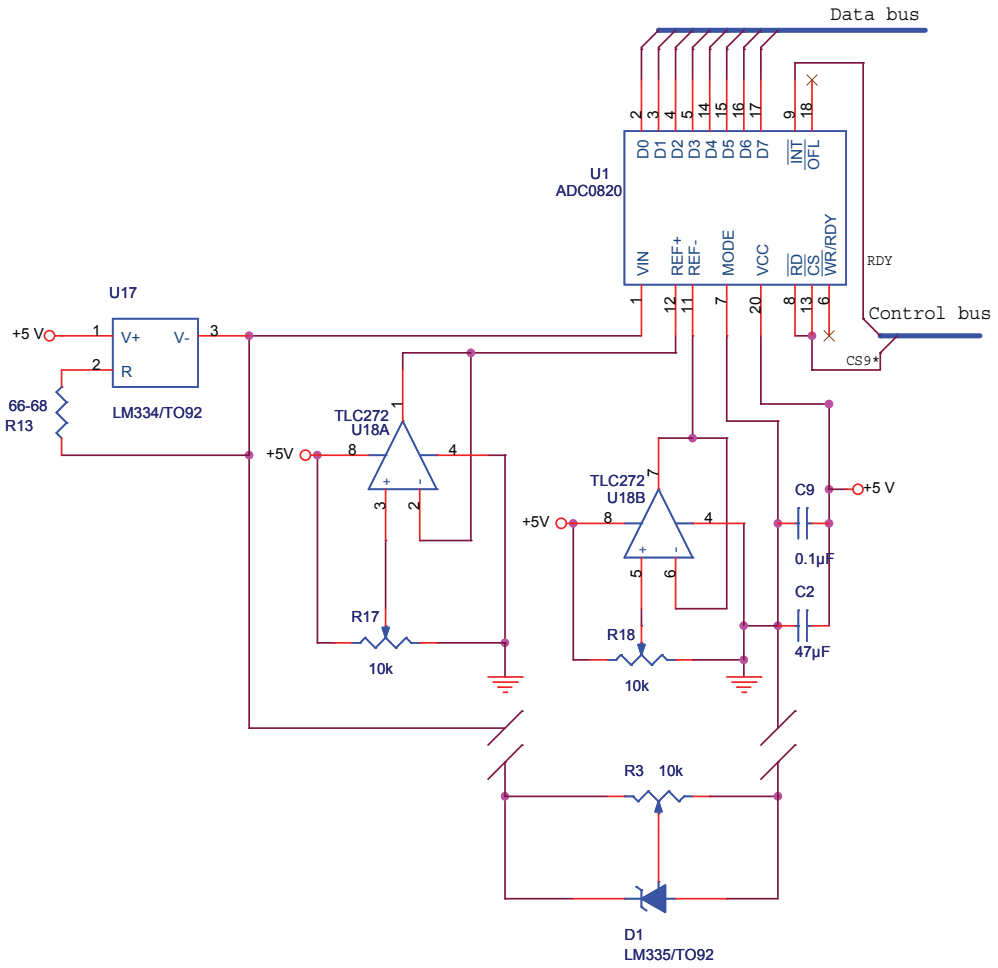

Figure 4.10: Circuit board of the digital temperature sensor module.

4.6.4 Summary

In this section the circuitry for the ADC and the temperature sensor has been described. The ADC and temperature sensor acts as a digital thermometer, by using the output from a LM335 and converting it into a binary value, which can be read by the m68k.

4.7 Liquid Crystal Display

In this section the design of the LCD will be described. The function of the LCD is to enable the user to receive information, regarding system status and administrator/user information. In relation with a numeric keypad [6], the LCD will make up the local user interface, which allows the administrator/user to interact with the system.
The LCD used in this project is a standard type display, L1614 [3]. This display is produced by Seiko, and is capable of displaying sixteen characters on four lines, e.g. 64 characters each consisting of 5x7 pixels. The display also features an embedded controller, which facilitates the communication to the display.

The integrated RAM in the LCD is divided into two sections, the DD RAM and the CG RAM. In this project only the DD RAM will be used, since the project has no need for custom characters. The DD RAM contains the standard characters (ASCII defined), which are addressed using the display data bus. The LCD is able to operate either using a 4 bit or a 8 bit data bus. Since the M68k, [9], microprocessor data bus is 16 bit, the 8 bit configuration is used in this project. The remaining 8 bits on the microprocessor data bus, are in this case unused.

4.7.1 Location

The LCD is located on the address bus as a parallel peripheral. In figure 4.1 on page 35 the location of the LCD in relation to the M68k can be seen.

4.7.2 Functionality description

In this section the structure and functionality of the LCD is described in relation to the M68k. The first part of the section describes the pin connections and their functions. The second part describes the communication between the M68k and the LCD.

Pin connection

The internal circuitry of the LCD is illustrated on figure 4.11. The pin connections are described in table 4.9.

Pin	Description
DB0-DB7	Data pins
E	Data input enable, - used as strobe. When E is low no data is read of the address pins. When E is high the display is open for communication.
$R/\overline{W}$	Read/Write. When $R/\overline{W}$ is low the display is in write mode and when it is high it is in read mode.
RS	Register Select. When RS is low the display is open for introduction commands[1]. When RS is high character can be sent to the data register.
VDD	Positive supply voltage pin. (5V)
VLC	Display contrast pin. (effectively connected to a voltage divider).
VSS	Ground.

Table 4.9: Pin connection for the display.

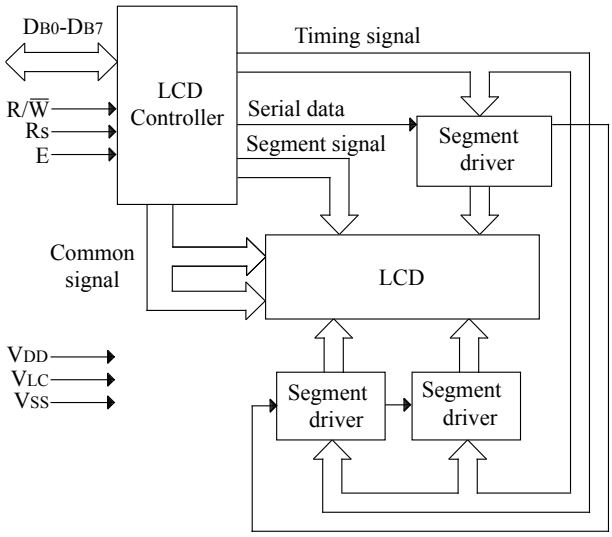

Figure 4.11: Circuit block diagram for the display.

Communication

This section describes the typical write cycle for the LCD. In this example the 8 bit data bus is used to communicate with the M68k.
The objective of the write cycle is to display a menu type text. Note, that this menu is only an example and has no real functionality.
When communicating with the LCD, there are a number of different options, that enable the programmer to control the display of the characters and the state of the display. These options are described in table 4.12.

Instruction	Code										Function	Execution time **
	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0		
(1) Display Clear	0	0	0	0	0	0	0	0	0	1	Clears all display and returns cursor to home position (address 0)	1,64 ms
(2) Cursor Home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position, shifted display returns to home position and DD RAM contents do not change	1,64 ms
(3) Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets direction of cursor movement and whether display will be shifted when data is written or read	40 µm
(4) Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Turns ON/OFF total display (D) and cursor (C), and makes cursor position column start blinking (B)	40 µm
(5) Cursor/Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves cursor and shifts display without changing DD RAM contents.	40 µm
(6) Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), the duty (N), and character fonts (F)	40 µm
(8) DD RAM Address Set	0	0	1	ADD							Sets DD RAM address to start transmitting or receiving DD RAM data	40 µm
(10) Data Write to CG RAM or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM	40 µm t_{ADD}=6µm
(11) Data Read from CG RAM or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM	40 µm t_{ADD}=6µm

Figure 4.12: Command interface in relation to the LCD.

When text is to be shown on the display, the microprocessor sends a series of hexadecimal codes to the LCD's address bus, after initialization. The initialization can contain several stages, but as a minimum consists of the following three steps described in table 4.10.

Step name	**Description**
Display sequence	The display is reset and forced into a known state
Entry mode set	Determine the way the characters are incremented on the display.
Function set	Determine the used data length (8 bit) character font and number of display lines

Table 4.10: Minimum initialization steps for the LCD.

Whenever the display is turned on or reset, it must be initialized to ensure correct operation and known settings[2]. The initialization is accomplished, when using the 8 bit data bus, by sending the hexadecimal code 0x38 to the data bus multiple times. This is done in order to ensure recognition of the code.
The objective of the next step is to ensure, that the display is in the desired mode. This code string enables

[2]The initializing turns on the cursor and clears the display. The display is also forced into an auto-increment mode

the programmer to set the interface data length, the number of display lines and the display font. Table 4.13 indicates the code sent to the address bus, when initializing function set.Setting DL high, sets the interface data length to 8 bits, i.e. the 8 bit address bus is used in total length. N indicates the number of lines used on the display. F sets the used font size, i.e. if F is set high a 5 x 10 dot matrix is used and if F is low a 5 x 7 dot matrix is used.

The following example describes the write cycle involved, when displaying the pseudo menu in table 4.11 on the display.

MENU
Temperature set
Alarm set
Status

Table 4.11: Pseudo menu.

A description of the data flow is illustrated in figure 4.13. The figure describes the reset sequence, the initialization and the writing of the word "MENU". The reason, that only the "MENU" text is written, is that the rest of the code follows the same configuration. For an elaborate description of the hexadecimal code sent to the display, when writing, see the software section concerning the locale user interface 5.7 on page 84.

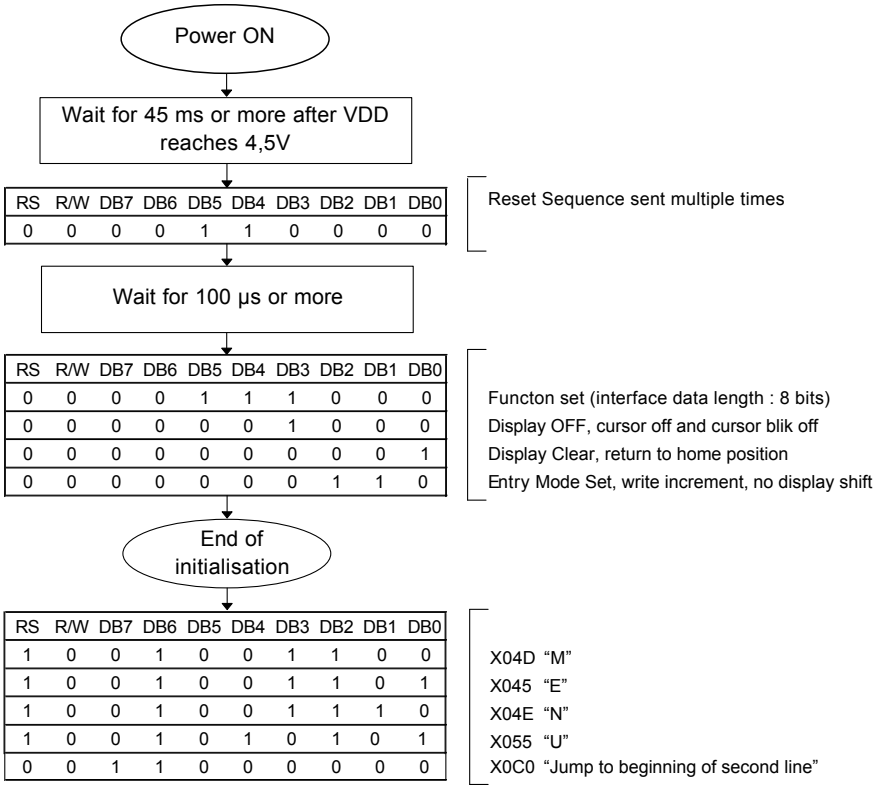
Power ON
Wait for 45 ms or more after VDD reaches 4,5V
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 0 0 0 0
Reset Sequence sent multiple times
Wait for 100 µs or more
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1 1 0
Functon set (interface data length : 8 bits)
Display OFF, cursor off and cursor blik off
Display Clear, return to home position
Entry Mode Set, write increment, no display shift
End of initialisation
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
1 0 0 1 0 0 1 1 0 0
1 0 0 1 0 0 1 1 0 1
1 0 0 1 0 0 1 1 1 0
1 0 0 1 0 1 0 1 0 1
0 0 1 1 0 0 0 0 0 0
X04D “M”
X045 “E”
X04E “N”
X055 “U”
X0C0 “Jump to beginning of second line”

Figure 4.13: Write cycle for displaying the pseudo menu.

4.7.3 Circuit design

In this section the physical design of the LCD in relation with the M68k is described. In figure 4.14 the circuit design of the LCD in relation with the M68k is illustrated.

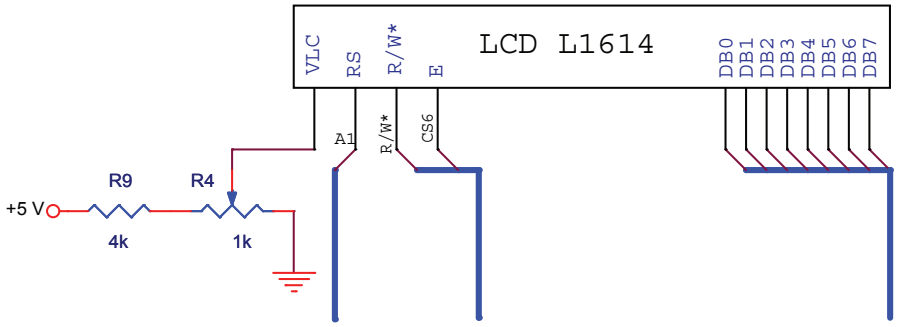

Figure 4.14: Circuit design of the LCD in relation with the M68k.

In order to communicate with the LCD, the system needs to have a series of control lines. The control lines consist of the RS and R/$\overline{W}$ and a data enable E. The data enable is used as chip select and ensures, that data is not read from the address bus, before the data enable is asserted. This prevents the display from accidentally reading data from the address bus. The M68k has a R/$\overline{W}$ pin that indicates whether the unit is reading from the data bus or writing to the data bus. This functionality is utilized to control the R/$\overline{W}$ pin, that indicates, whether data is being read from or written to the display.
When initializing the display, it is necessary to know whether the display is busy or not. This is achieved by using the busy flag (BF) pin, that indicates whether the display is busy with internal operation or ready for communication. Table 4.12 describes the BF read operation; when BF is high, the module is working internally and the next instruction cannot be accepted until BF goes low.

	Rs	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	1	BF	*	*	*	*	F	*	*

Table 4.12: Busy Flag read. $*$ = don't care.

Display contrast

In order to adjust the contrast of the display, the VLC pin is connected to a voltage divider. In figure 4.14 the setup is illustrated. The used voltage divider consists of a variable resistor (R4) and a resistor (R9) between GND and positive voltage supply 5 V. The used variable resistor is a $1K\Omega$ and the resistor is a $4K\Omega$, that gives the following theoretic voltage interval; - see Eq. 4.24.

$$VLC_{min} = V_{cc} \cdot \frac{R_{min}}{R_{min} + R_{max}} \tag{4.20}$$

$$= 5 \cdot \frac{0}{4k + 1K} = 0V \tag{4.21}$$

$$VLC_{max} = V_{cc} \cdot \frac{R_{max}}{R_{max} + R_{min}} \tag{4.22}$$

$$= 5 \cdot \frac{1K}{4K + 1k} = 1V \tag{4.23}$$

$$VLC_{min,max} = [0 - 1][V] \tag{4.24}$$

4.7.4 Summary

In this section the design of the LCD L1614[3] in relation with the M68k has been described. The typical write cycle for the display has also been accounted for through an example.

4.8 Keypad

In this section the design of the keypad is described. The function of the keypad is to enable the administrator/user, to enter data into the system. Together with the LCD, section 4.7, the keypad make up the local user interface, which allows the administrator/user to interact with the system.
The keypad used in this project is a standard keypad, that consists of matrix arranged pressure sensitive switches. This keypad is used because of its low cost and slick design. The used keypad is a self-contained device, which holds more buttons, than necessary in this project. This is however no problem, since the constructed device is a prototype, which in a commercial situation will be simplified. The used matrix setup enables, that the keypad can be replaced with as few switches as needed in a commercial situation. In figure 4.15 the general design of the keypad is illustrated.

Figure 4.15: A keypad similar to the one used in the GSM RCS.

In order to use the keypad in relation with the M68k. The keypad signal is controlled by a 4 bit encoder. The used encoder is a 74C922 [2] that changes the input from the 12 pressure sensitive switches to a 4 bit binary value.
The reason, that an encoder is used is, that it saves room on the data bus and it features an embedded data available signal (DAV). This signal is used with the interrupt encoder, to generate the interrupt for the keypad. The encoding ensures a far more manageable handling of the data flow from the keypad and implementation of the interrupt signal.

4.8.1 Location

The keypad is located on the M68k data bus as a parallel peripheral. In figure 4.1 on page 35 the location of the keypad is illustrated.

4.8.2 Functionality description

The keypad is arranged in a matrix as illustrated on figure 4.13. As illustrated, the keypad consists of 3 by 4 switches. Theses switches are divided into three columns and four rows. When a button is pressed, it creates a path between a column and a row. The used encoder is specifically designed for a 4 by 4 matrix

not a 3 by 4 matrix. This is however not a problem, since the inputs assigned for the fourth column are simply unassigned.

	X1	X2	X3	X4
Y1		1	2	3
Y2		4	5	6
Y3		7	8	9
Y4		*	0	#

Table 4.13: The matrix setup for the keypad.

Entry mode

When the keypad is in static mode, i.e. when no button is pressed, the row inputs are pulled high and the column outputs logic, low. The scan rate, by which the columns are scanned, is controlled by an oscillator. When a button is pressed, e.g. button 1, nothing will happen, when the X2 input is off, since Y1 will remain high. When the X2 column is scanned, X2 goes low and Y1 will go low. The low signal on Y1 interrupts the counter and keeps X2 low.
When Y1 goes low, it also initiates the key bounce circuit timing and in reality blocks other Y inputs. The pressed switch is now outputted as a combination of the frozen counter value and the decoded Y input. Once the key bounce circuit times out, the data is latched, and the DAV output goes high. To ensure, that data is not placed on the data bus at random, the 74C922 encoder [2] features an output enable ($\overline{OE}$), that determines, when the data is put on the data bus. This is another advantage of using the 74C922, since the $\overline{OE}$ can be used as chip select. When buttons are used in digital circuits, there is an imminent risk that the buttons will bounce[3]. If the pressed switch bounces during the switch closure, Y1 input will go high again, restarting the scan and resetting the key bounce circuit. The switch can bounce several times, but only if the switch stays low for a period, the closure is considered valid.

4.8.3 Circuit design

The encoder features four data pins and a series of control pins. The four data pins are simply connected to the M68k data bus. The control pins, consist of a DAVBL (Data Available) and $\overline{OE}$.
In order to ensure a secure data transfer, the keypad is designed to support synchronous handshake operation. To achieve this operation mode a D-flip-flop is implemented, as shown in figure 4.16. The result of this configuration is, that when a button is pressed, the data available goes high. Whenever the M68k is ready to handle the data, the encoder is chip selected. At this point the data is put on the data bus.

[3]The switch produces several logic high signals even if only pressed once. The effect is caused by the physical design of the switch

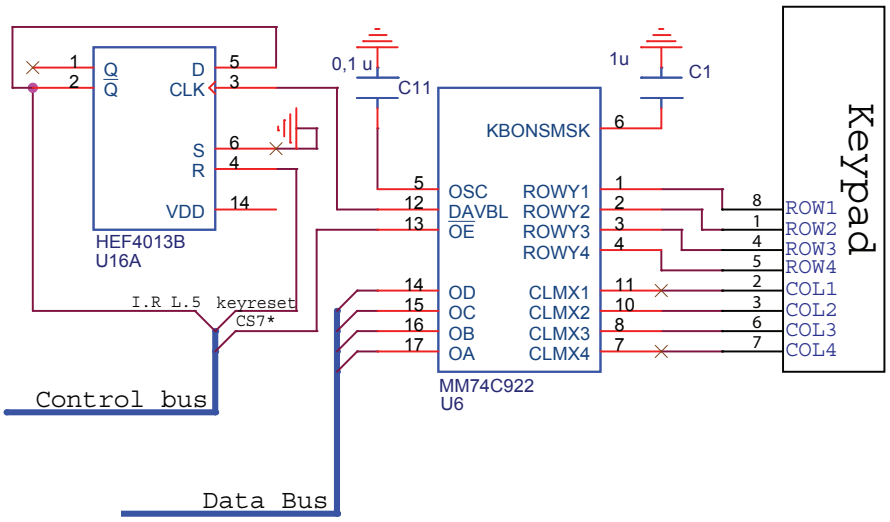

Figure 4.16: Keypad circuit design.

In order to control the scan rate and the debounce period, two capacitors are connected to the encoder. The capacitor C_{OSC} controls the scanrate (F_{scan}) and C_{KBM} controls the debounce period. In accordance with the datasheet the values $C_{OSC} = 0.1\mu F$ and $C_{KBM} = 1.0\mu F$ are chosen. This results in a scanrate $F_{scan} = 1500Hz$ and a debounce period $C_{KBM} = 0.01s$.

4.8.4 Summary

In this section the design of the keypad has been described. The 3 by 4 keypad is accessed using an encoder, that enables synchronous handshake operation.

4.9 Mock system

In this section the design of the mock system will be described. The function of the mock system is to simulate a consumer product that is to be controlled by the GSM RCS. In reality the mock system could consist of various products that could be turned on and of by the system. However, the main focus of the project is to control a specific device, but to show the functionality of a system capable of doing so. As a consequence the mock system consists only of a switch with no real functionality other than switching a LED on and off.

4.9.1 Location

The mock system is located on the address bus as a parallel peripheral. On figure 4.1 on page 35 the location of the mock system is illustrated.

4.9.2 Functionality description

The objective of the mock system is to enable the control of a LED using only one bit on the address bus A_1 and a chipselect. The use of only one control line give rise to two possible static states logic 1 and 0. The circuit must however, be able to remember the state of the control line, as it is only strobed

during initialization. This memory type functionality is achieved using a D-flip-flop. The mock system is constructed as illustrated on figure 4.17. The clock (CLK) is strobed when configuring the mock system and the truth table in table 4.14 is achieve. The table is consistent with the desired functionality of the unit, where the state of the control line D is kept as output Q even when D returns to a different state. It is noticed the state of Q before the strobeing is irrelevant.

Pin	Control line (address bus A_1)	Q (output)	Q_{next}
State	1	X	1
State	0	X	0

Table 4.14: The truth table for the mock system.

When the system is to turn the LED on or off the control line (Address bus pin A_1) is set high or low, dependent on the desired action. Subsequent the clock i.e. chipselect is forced high and the state of the control line becomes the output Q.

4.9.3 Circuit design

In figure 4.17, the circuit is illustrated. The used LED L53LI [5], is a small 3mm low current LED. Using this type of LED, allows the LED to be driven directly from the flip-flop, without any current amplification. The CMOS 4000 family can source about 5mA, when supplied by 5V and the LED consumes 2mA with a forward voltage drop of 2V. In order to limit the current to the LED, a limiting resistor R_{LED} is implemented in series with the LED.
The size of the resistor, R_{LED}, used to limit the current through the LED, is calculated using Eq. 4.25. Note, that V_s is the source voltage from the flip-flop and V_F is the forward voltage drop of the LED. I_{LED} is the recommend current through the LED at V_F = 2.0V.

$$R_{LED} = \frac{(V_s - V_F)}{I_{LED}} \tag{4.25}$$

$$= \frac{(5V - 2.0V)}{2mA} \tag{4.26}$$

$$= 1.5K\Omega \tag{4.27}$$

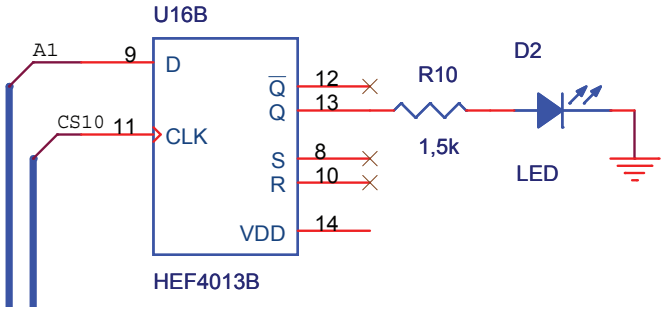

Figure 4.17: Circuit design of the mock system.

4.9.4 Summary

In this section the design of the mock system has been described. The unit consists of a small logic gate configuration and a LED. The mock system uses the address pin A_1 and chip select as primary control lines.

4.10 Address Decoding

To address specific modules in the system, it is necessary to decode addresses from the M68k, to generate chip select. The address range, which is reserved for the RAM, ROM and the UART has already been discussed. The additional peripherals of the system, the LCD, keypad, temperature sensor, GSM module and the mock system, is added into the design of the chip select.
The usage of a truthtable helps to identify the different modules, that are to be addressed and how to generate the correct chipselect. In table 4.15, the truthtable for the chip select is illustrated. Using Boolean algebra on the truthtable, the chip select logic can be expressed as illustrated in table 4.16. Notice, that the pins marked X_x in table 4.15 are not "don't care" in the common sense as one of the X_x will always be one.

Module	A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17-2}	A_1	LDS	UDS	$\overline{AS}$	CS
ROM1	0	0	0	0	0	0	X	X	0	1	0	0
ROM2	0	0	0	0	0	0	X	X	1	0	0	0
RAM1	0	0	0	0	X_x	X_x	X	X	0	1	0	0
RAM2	0	0	0	0	X_x	X_x	X	X	1	0	0	0
UART	1	0	0	0	0	0	0	X	1	0	0	0
LCD	1	0	0	1	0	0	0	X	1	0	0	1
Keypad	1	0	1	0	0	0	0	X	1	0	0	0
GSM	1	0	1	1	0	0	0	X	1	0	0	0
Temp. Sensor	1	1	0	0	0	0	0	0	1	0	0	1
Conrtrol unit	1	1	0	1	0	0	0	0	1	0	0	1

Table 4.15: Truthtable for addressing of ROM, RAM, and all the peripherals. Values 1 or 0 are the required state on the address bus, X's are "don't care"'s.

ROM1:	$A_{18}+A_{19}+A_{20}+A_{21}+A_{22}+A_{23}+UDS$
ROM2:	$\overline{A_{18}+A_{19}+A_{20}+A_{21}+A_{22}+A_{23}}+LDS$
RAM1:	$A_{23}+A_{22}+A_{21}+\overline{(A_{20}*\overline{(A_{19}+A_{18})})+(\overline{A_{20}}*(A_{19}+A_{18}))}+UDS$
RAM2:	$A_{23}+A_{22}+A_{21}+\overline{(A_{20}*\overline{(A_{19}+A_{18})})+(\overline{A_{20}}*(A_{19}+A_{18}))}+LDS$
UART:	$\overline{A_{23}}+A_{20}+A_{21}+A_{22}+VMA$
LCD:	$A20*\overline{A21}*\overline{A22}*A23*\overline{LDS}$
Keypad:	$A22+A20+\overline{A21}+\overline{A23}+LDS$
GSM:	$A22+\overline{A20}+\overline{A21}+\overline{A23}+VMA$
Temp. Sensor:	$A21+A20+\overline{A22}+\overline{A23}+LDS$
Controlunit:	$\overline{(A21+\overline{A20}+\overline{A22}+\overline{A23}+LDS)}$

Table 4.16: Logic equations for generating Chipselect signals.

4.10.1 PEEL

In addition to addressing, it is also necessary to connect the VPA and VMA pins to the GSM module. Since the PEEL chip, 22CV10A [4], has 10 I/O-pins and 13 dedicated input pins, two PEEL chips are required to be able to generate the needed $\overline{CS}$ signals. The logic description of the $\overline{CS}$ signals for the peripherals can be seen in table 4.16. The $\overline{CS}$ signals for the LCD and the Control unit are active high, the rest is active low. The address decoding and the necessary operating signals for these, can be generated with the first two PEEL's, but a third PEEL is needed to generate interrupt signals and handle interrupt acknowledge signals from the M68k. The raw ABEL code for the third PEEL can be seen in appendix G.3.

4.11 Microprocessor Communication

The M68k is able to transfer data, between itself and the memory in two ways. The data transfer can happen either synchronously or asynchronously. This section contains a short summary of what each of these data transfers are, and a discussion of the chosen bus control. Unless otherwise specified, the source is from reference [1].

4.11.1 Synchronous Bus Control

When using synchronous data transfer, the m68k provides an address and a strobe or clock to indicate that the address is valid. Ideally, a read cycle begins at the falling edge of the clock, where afterward the M68k generates an address to the memory location, where it is supposed to access data. After the address has been generated, the data is provided to the M68k. The read cycle ends at the falling edge of the clock. The time between the data is provided and the end of the read cycle is called the M68k's data setup time. A this point the data must be valid. An illustration of the ideal synchronous bus control is shown in fig. 4.18.

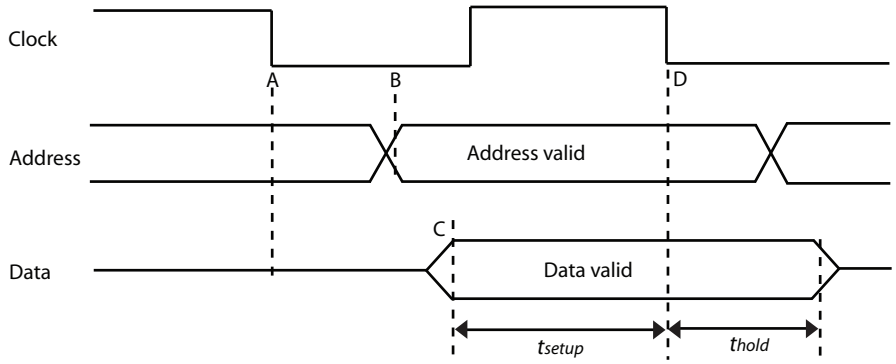

Figure 4.18: Idealized timing diagram of synchronous data transfer of the Motorola 68000 CPU. [1]

The synchronous bus control is designed to simplify the interface between the M68k and peripherals used with older 8 bit synchronous bus microprocessors. All these peripherals are easy to interface with a M68k bus. However, they are not simple to interface with the asynchronous interface, although not entirely impossible. The M68k has been provided with a synchronous bus control, to permit designers to use low-cost and already tested, peripherals. Because of this, the M68k has been provided with three bus control signals.
In the synchronous interface there are three control signals, $\overline{VPA}$ (Valid Peripheral Address), $\overline{VMA}$ (Valid Memory Address) and E (Enable).

The $\overline{VPA}$ is used when a peripheral detects, that it is being accessed. The peripheral asserts the $\overline{VPA}$ to indicate that a bus cycle is requested. When the processor has recognized, that the $\overline{VPA}$ has been asserted, a synchronous data transfer is initiated by the $\overline{VMA}$ signals.
$\overline{VMA}$ is used to inform the peripheral, that there is a valid address on the address bus. When $\overline{VMA}$ is asserted, it is a response from the M68k of the assertion of $\overline{VPA}$ by an addressed peripheral.
The enable output comes from the M68k's own clock input and is a timing cycle required by all 6800-series peripherals.

4.11.2 Asynchronous Bus Control

Asynchronous Bus Control is more complex than its synchronous counterpart. The bus can be run asynchronously, to achieve an independency of the clock frequency. Devices, that can run asynchronously, are designed to operate correctly with processors with no regard to the clock frequency. It uses the bus handshake signals to control the data transfer. These handshake signals consist of $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, $\overline{DTACK}$, $\overline{BERR}$, $\overline{HALT}$, and $\overline{VPA}$.[9]
$\overline{AS}$ indicate when the bus cycle begins, and the $\overline{UDS}$ and $\overline{LDS}$ signals validate the data for a write cycle. Afterward, $\overline{DTACK}$ is asserted to terminate the buscycle, when the requested and latched data is put on the databus.

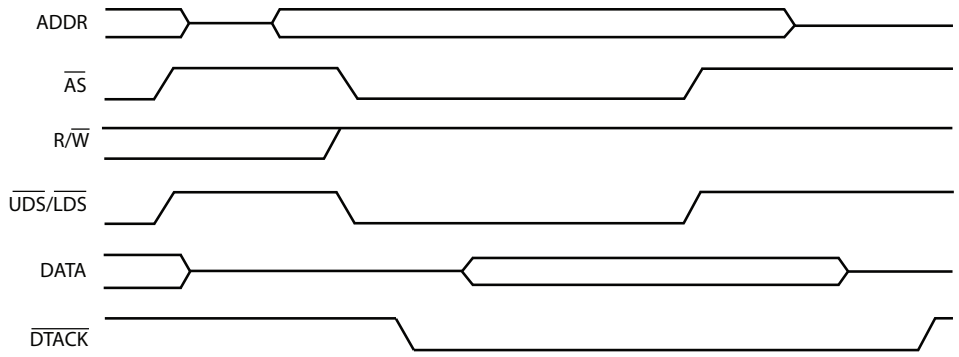

Figure 4.19: Asynchronous Read Cycle. [9]

A general procedure of how the asynchronous operation of a M68k, begins with the processor generating an address and asserts the address strobe. This informs the memory, that the generated address is valid. When the memory detects, that the address strobe is asserted, the data is placed on the databus, and becomes valid. Afterward, the memory asserts its data acknowledge strobe to indicate to the processor, that the data is valid. The processor detects the strobe, reads the data, and negates the address strobe, to indicate, that the processor has read the data. Subsequently the memory negates the data acknowledge signal to complete the cycle. This can be seen in the timing diagram in fig. 4.20, which resembles the diagram in fig. 4.19.

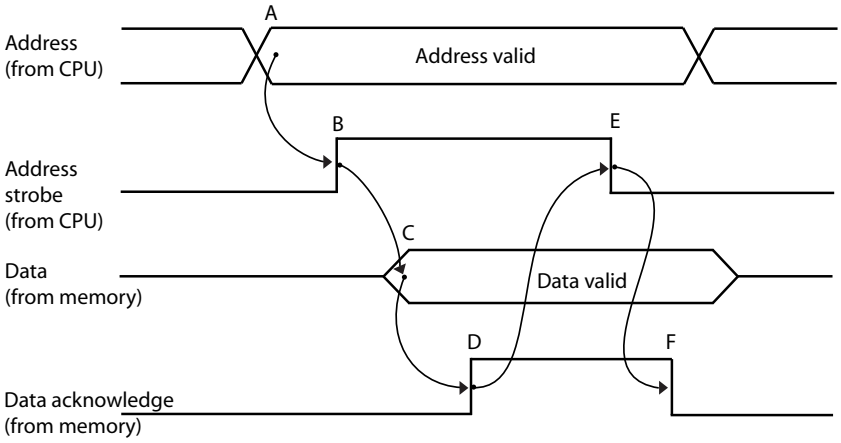

Figure 4.20: An idealized asynchronous data transfer in a read cycle. The control signals are active-high. [1]

4.11.3 Summary

The ACIA peripherals are running synchronously with the CPU, while everything else runs asynchronously. The reason for this, is that the ACIA peripherals are the only ones that are not as fast or faster than the M68k.

4.12 Implementation

After designing the circuits and the wrap diagram, the board layout can now be designed. The electronics has been fit close together for two reasons. The first reason is that the circuits, LCD and keypad can be fitted on a single plane circuit board. Secondly a tight circuit placement also ensures short wiring, which minimizes problems caused by reflections of digital signals in the wiring.

4.12.1 Reflections

Reflecting signals occur, when a digital pulse travels though a wire, which is not impedance adapted. This can cause the signal to reflect a fraction of the sent signal back, when it arrives at the receiving end of the wire. In worst cases, the reflected signal will either cancel out the incoming signal, or resonate with it, effectively doubling the voltage at the receiving end. In the first case, no signal will be detected, preventing the circuits to perform as expected, the second case will most likely cause the receiving chip to be overloaded.
Reflections happen when a wire reaches a certain length, depending on the signals wavelength. This wire length is quite long for the signals being transmitted in this project. Since the design ensures very short wiring, the problem will not be addressed further.

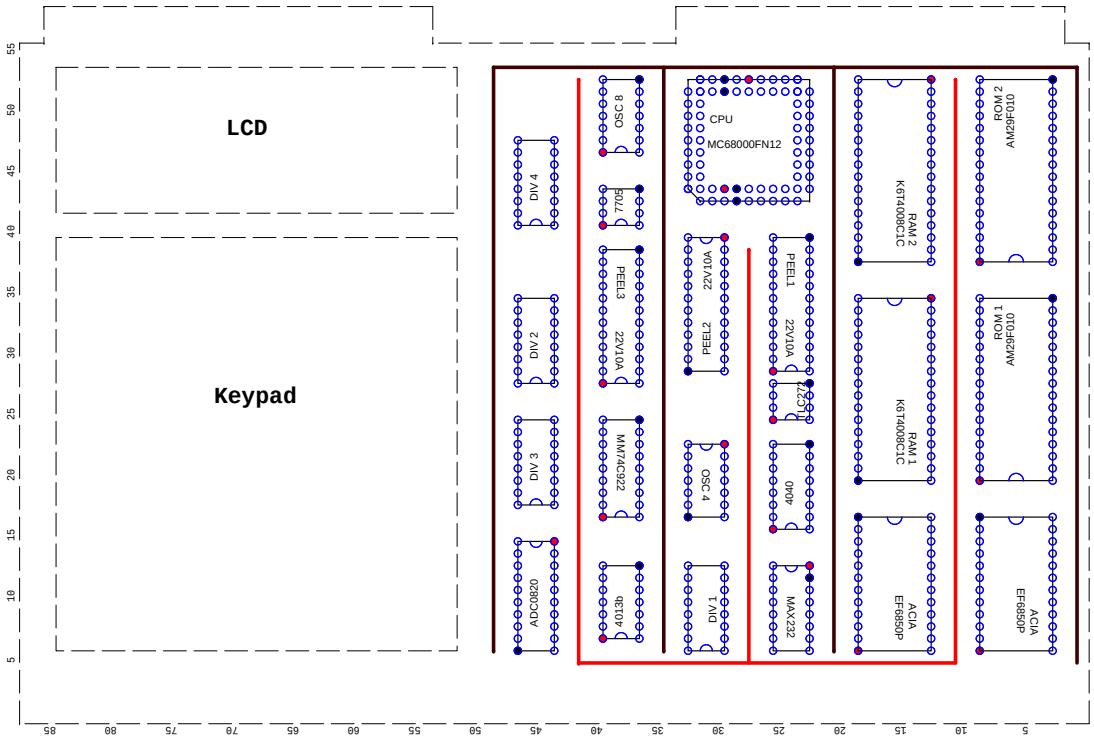

Figure 4.21: Placement Diagram seen from above. The thick black lines is the ground line, black pins mark the ground pins on each chip. The thick red line is the 5 V supply line, red pins mark V_{CC} pins on the chips.

4.12.2 Ground and supply lines

As seen in figure 4.21, there is a main ground wire and a main supply wire, which branches into several lines between the chip sockets. The sockets have been placed, so that each sockets ground and supply pin, is as close as possible to their respective line. Both the ground and supply line has one connection each to an external power supply, which delivers the ground signal and the 5 V signal.
The ground and supply lines are both mounted on the top side of the circuit board. The lines are soldered on top of several wrapping pins, that are strategically placed, to complement as short wiring as possible. This placement serves another purpose, as each chip may strain both the ground and the supply line, when active. Connecting too many chips to the same point on the ground or supply line, may disrupt the functions of all the chips connected to this point. By having several outtakes along the ground and supply lines, and distributing the chips connected to them evenly, the strain each chip applies should have a minimal effect on the overall functionality of the circuitry.

4.12.3 Noise prevention

The circuitry is vulnerable to noise. This is because of the high frequency the chips operate at. The high frequency means, that the logic levels of the digital signals, must be well defined to be correctly interpreted. The signals must fulfill two criteria to be correctly interpreted. First of all, the voltage level must be over a certain level, to be interpreted as high and below a certain level to be interpreted as low. In addition to this, the edges of each pulse must also be distinguishable, since several of the chips use the rising and falling edges, when executing functions.

To accommodate these demands, each chip will first of all need to have stable supply and ground levels. To prevent fluctuations on the power line affecting the circuits, bypass capacitors are used. These capacitors will short most alternating currents to the ground line.
Unfortunately, one of the circuits used, generates enough noise to locally disrupt the power or even ground line. This is the reason why the location of the MAX-232 chip, and especially its bypass and decoupling capacitors, have been placed as far as possible from the most sensitive chips.

4.12.4 Practicalities

RAM and ROM

For clarity, the chips with many connections to the address bus and the data bus, RAM and ROM, have been grouped on the right side of the board. In this way, the wires for the two buses, will not be crossing over any large areas of the circuit board, making testing and troubleshooting easier.

PEEL's

PEEL 1 and 2 both use several inputs from the address bus and both primarily generate chip select signals. PEEL 1 generates chip selects for both RAM and ROM, and is therefore placed closest to these. PEEL 2 generates chip selects for the peripheral devices, which are spread over most of the board. Since PEEL 2 takes inputs from the address bus, it should still be placed close to the M68k. The third PEEL, PEEL 3, generates interrupt signals, both for the processor and for PEEL 2. It must like PEEL 2 be placed close to the M68k.

Miscellaneous

The sockets marked DIV 1, 2 and 3 are sockets used to mount external components needed for some of the chips. They are used for larger coupling, decoupling, bypass capacitors, diodes, pull up and pull down resistors. Div 1 is dedicated to MAX-232 components due to the noise generated in these.

Chapter 5

Software Design

In this chapter the basic buildup of the system software, the organization of the subprograms, the design and the functionality of the software is described.

5.1 Design procedure

The chapter is based on the SPU model, that separates the design process into three sub-processes as illustrated in figure 5.1. Using the SPU model, allows the logical branching in relation to the functionality of the system.

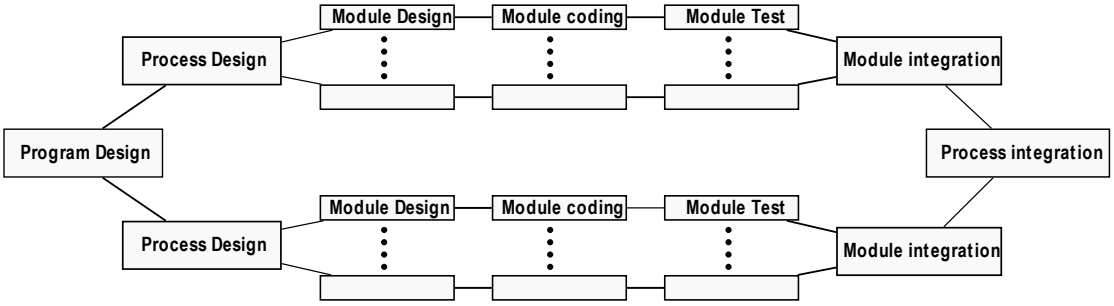

Figure 5.1: The sub-processes of the software design process.

The basis of the SPU is a pyramid like structure, that enables the programmer to maintain an overview of the design process. The structure also divides the overall program into smaller functions. These functions are then divided into even smaller functions, ending up with a pyramid structure. This allows the branching of a complex program into manageable tasks. The task syntax in the process also becomes well defined, which gives room for outsourcing.

In the program design section, the overall functionality of the system is described, using flowcharts and task handling. The overall functionality is divided into sub-processes, that narrows down the tasks, in order to arrange the processes in modules. The individual module objectives are described and the modules are treated as individual programs. Similar to the program design, the external interface of the modules is described. The module design contains a detailed description of the data structures, that are contained in the modules. The description is based on both pseudo-code and actual assembly code.

5.2 Program structure

There are several ways to structure the execution of the systems functions, and there are multiple options to consider, when programming the system. One of the things to consider, is the way the program code is structured. This not only gives an organization of the code, but can also be reflected directly to the execution of the software. In this section, the basic program structures and their functionalities are described. The objective of the section is to identify the most effective program structure, for designing the system software.

Stand-alone Main program

In this model, the program runs in the same main function continuously, from beginning to end. If the program is structured in this way, it is important to take special notice, of which variables are used as global and local variables. This is important since the different functions may use the same variables more than once.
This style of programming, will only be able to handle specific actions, at a specific location in the program. This means, that there will be no outside influence on the program execution, unless the program is set to pull data at a predefined time and/or condition specified by the programmer.

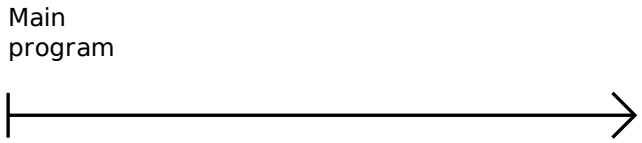

Figure 5.2: Programming done in a single line.

Standard Subroutines

Using subroutines in a program, can give a more organized code structure. Subroutines work as a part of a program, which can be called from the main program. This gives several advantages. Creating subroutines, enables the programmer to write the software in various programming languages. This can be an advantage when complex programs are designed. Furthermore, using subroutines gives a procedure in the execution of the program, that makes it ideal to use the stack to store the used variables. The concept of the stack is addressed in section 5.2.1.

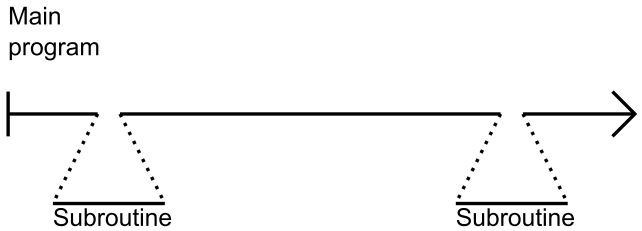

Figure 5.3: Program spiced up with subroutines.

Standard Interrupts

Interrupts works in the same manner as a subroutine, except it is initiated by an external signal. This can for an example, be an I/O device, that needs to be managed. Each interrupt is given a certain level related to how important it is. For example input from a keypad, would have a relatively high masking level. This is done to ensure that the task is attended to as fast as possible. If the system has relatively few interrupt functions, it is possible to use auto-vectored interrupts. This means, that every time a specific interrupt is triggered, a certain piece of code is run. The program can only contain 7 interrupts when using auto-vectored interrupts on the M68k.

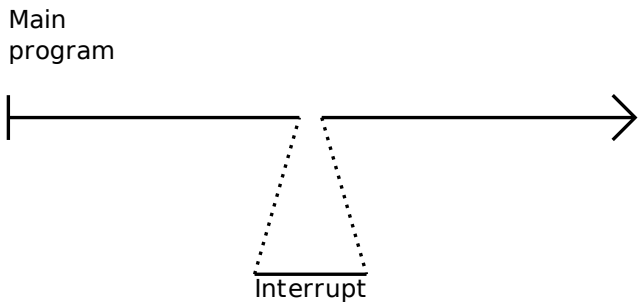

Figure 5.4: Hardware interrupt incorporated.

Combined program structure

By combining subroutines and interrupts, the main program can be a relatively small loop, where subroutines only will be executed, when the conditions are right. This structure also allows the I/O devices to interrupt the main program.
By minimizing the main program, the number of local variables, is reduced, since the subroutines still have to communicate. Using the internal registers of the processor, free variables and incorporating a series of global variables will, however, simplify the handling of the variables. As a consequence of this, the stack will not be used much, which further simplifies the flow of the program.

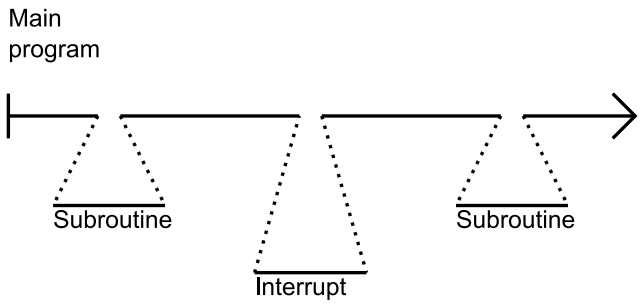

Figure 5.5: A program structure as uses multiple subroutines and interrupts.

5.2.1 Stack

The stack is a convenient way to allocate dynamic workspace in a program. The stack is usually placed at the top end for the RAM memory, as in this project. The fact that the stack is dynamic means, that it has no predefined size, but can expand downwards in memory to meet the requirements of the program. The

size limit for the stack is determined by the amount of memory, used for the program and global variables, which is placed at the beginning for the RAM.

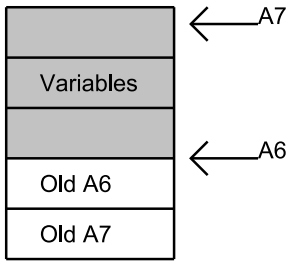

Figure 5.6: A example of a stack where each box is a word.

The stack uses 2 addresses or pointers to keep track of the current stack. These addresses are stored in the internal registers of the processor, A6 and A7. A7 points to the highest end of the stack and A6 at the lowest end. The size of a stack is determined, when the stack is initialized and by the subroutine, that grants it. When a stack is created, the pointers are moved to the new address of the stack, while the old values of the registers are placed in the memory, just before the new stack. This ensures, that when a subroutine is finished, the stack pointers can return to the old stack. Because the old pointers are saved, it is possible to run a subroutine inside another subroutine. This is an advantage, when working with interrupts, where it is not sure that the processor is finished with a subroutine, when the interrupt is activated.

5.2.2 Summary

To get an organized structure in the system software, a relatively small main program is used. The system is based on the implementation and the use of subroutines and interrupts. This makes the program more flexible and allow room for implementation of further features. The program structure requires quite a large amount of global variables, but since the functionality of the system dictates large amounts of data, the use of the stack is mostly omitted. This is done, because the action of pushing and popping the large amounts of data onto the stack every time a subroutine is run, is considered more inefficient and time-consuming, than using large amounts of variables.

5.3 Storage structure

Before any actual code is written, the division of the memory space is identified. This is done to ensure that there are no code, variables or data, that conflict with each other. Furthermore, by defining the data space before programming, will ensure that the programming of the different subroutines, can be divided between different group members, without the danger of overwriting data or storing data multiple times at different locations.

5.3.1 Use of internal registers

Some of the internal registers are reserved for specific data and contain either single or several variables. Since the registers are available to all subroutines, all the registers can be temporarily used as long as backups are made and the registers are restored. The processor has 16 internal registers, 8 for data and

8 for addresses. The registers are desirable to use, since the processor can access them faster than the physical memory. The primary and predefined use of the different registers can be seen in table 5.1.

#	Data register	Address register
0	Free register	Free register
1	Free register	Free register
2	Free register	Free register
3	Menu counter	Free register
4	Status	Buffer pointer
5	Entry	Buffer pointer
6	Clock	Free register
7	Temperature	Stack pointer

Table 5.1: The internal registers.

- **Menu:** The register contains a number from 1 to 21, that indicates which menu the display is currently showing.
- **Entry:** The register contains a number from 0 to 9, that indicates the latest key pressed by the administrator/user.
- **Status:** The different bits in the register indicates which state the control and GSM module is in.
- **Clock:** This register contains the time. All four bytes are used for each digit in hours and minutes.
- **Temperature:** The register holds the temperature in ASCII characters. Example $+18.5°C$ is stored as +185 with "." omitted.

5.3.2 Program space definition

In order to avoid that the process codes overlap data and vice versa, the program space is divided between the two types of data (e.g. data and actual program code). In table 5.2 the division of the program space is illustrated. It is noted that the subroutines are placed in the same memory area. This is done as an attempt to bypass the problems, that can occur when branching between processes, address-wise, located far from each other.

Address	Module
0x41000-0x44FFF	Program code and subroutines
0x45000-0x45FFF	Data for LCD and GSM
0x46000-0x46FFF	Variables
0x47000-0x4712C	GSM buffer
0xXXXXX-0xBFFFF	The stack

Table 5.2: An illustration of the partition of the memory space in relation to the most significant datatypes and the actual program code.

String storage - Arrays

The system must store a number of text strings, that are used in relation to the menu subroutine, *MENU-MAIN* and the GSM routines. All the text strings are saved in arrays, to make them usable, all arrays ends with 0x00.

Data structure - Linked lists

In the software for the GSM module, there is need for a high amount of data, compeered to the other parts of the system. The main reason for this, is that the software needs to know all commands, responses and text commands from the administrator/user. In order to access this data, it is stored in a linked list. A linked list consists of a row of elements, each containing addresses for data and the address of the next element.

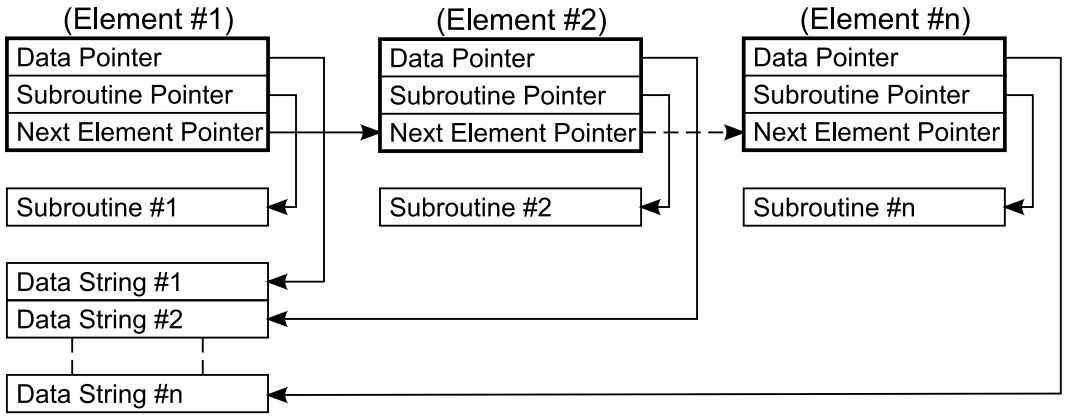

Figure 5.7: Illustration of a linked list.

In this project a linked list is designed, like the one illustrated in figure 5.7. Each element contains the address for a data string, that is identical to a response from the GSM module. Furthermore, there is an address for a subroutine designed to handle the specific situation. Each element also contains a pointer, that points to the next element. In this project all responses and user/administrator commands are stored in one linked list. This is done as it makes the list practical, as the same search function can be used in all situations. The list is not implemented to improve the performance and speed, as the search has to start with the first element and work its way through all elements, until it finds a match.

Temporary data storage - Buffer

The systems software is not designed to handle the data coming in from the GSM module, while it is being transferred. Therefore the data needs to be stored temporarily in the memory. This is done by using a buffer. A buffer uses additional space, but makes it possible to work with the data in multiple steps. This means that data does not have to be analyzed on the fly, i.e. while it is being sent by the GSM module. Two pointers located in the processors internal registers control the buffer. The first pointer points to the starting point of a string and other points to the end of the string. This means, that when the system is putting data in the buffer, the system uses one pointer and incrementing it for every character received. To ensure that the data does not slowly fill up the memory, the buffer is reset every time the data in it has been used.

5.3.3 TS2 debugger monitor

As shown in this section, all system software is placed in the RAM. This is due to the TS2 debugging monitor's requirements, as specified in section 5.4. The debugging monitor is used to test the software, without having to burn the ROM, before the software is in its final version. It is made clear, that all the software has to be loaded from an external source and stored in the RAM, before every single test. In the

final version of the software, the code and all static data will be placed permanently in the ROM memory, in order to make the system run without the debugger software. This means that all variables, that is not predefined and will be edited by the administrator/user or by the software itself, will have to be placed in the RAM.

5.4 TS2 Monitor

In order to debug the system software during development, a debugger system is used. The TS2 Monitor is a debugger/monitor program, that can be burned on the ROM of a MPU system. When the system boots, the TS2 monitor is initialized, which gives the programmer a range of features, without having any operating software.
The TS2 monitor allows the programmer to read and write to the memory modules and call any address on the address bus. This is very useful to test whether the chip selects are working correctly and whether the read/write functions is working for the RAM. The TS2 monitor also allows the programmer to upload code to the memory of the system and to run it from there. This a very useful function, when testing software and/or functionalities in the system.
When testing larger portions of software, it can be difficult to identify errors, when it is running real-time. Therefore, the TS2 monitor allows the programmer to add breakpoints to the software. The programmer can, through commands, add and remove break points in the code. When the processor reaches these breakpoints, the code will be "paused" and the TS2 monitor will allow the programmer to view the status register, internal registers and read memory addresses. Afterwards the code can be resumed. For more troublesome software errors, the TS2 monitor supports single stepping through the code. In this way, the programmer can see what happens for every step, and identify why the code is not running as intended.

5.5 Program design

In this section the overall functionality of the system will be described. The objective of the section is to divide the system program into sub processes.

5.5.1 Program overview

In accordance with the project analysis on page 3, the system contains several participants, that can influence the state of the system. These participants and their interactions form the basis of the system functionality. These interactions are used, when designing the system program, as the use cases can be transformed into program modules.
The program consists of a main program, that contains all the additional subroutines needed to obtain the desired functionality.

When the system is turned on, the main program calls the initialization subroutines *INIT_LCD* that initializes the LCD. The next routine that is executed, is the *MENUMAIN* that allows the administrator to enter the information, necessary to initialize the GSM module. The next routines *INIT_GSM* and *INIT_BUFFER* manages the initializing of the GSM module and the buffer.
After the initialization has been executed, the main program calls the normal operation routine *NORMAL*, that is running continuously. The normal operation routine, contains a minimum of actual code, but calls several subroutines that manage different tasks. An example of such a subroutine could be the *ADC*, that manages the update of the measured temperature.
The main program can be interrupted by interrupts of different levels in order to manage different tasks. The interrupts force the main program to go into exceptions, that contain functions related directly to the

interrupt in question.
There are two auto vectored interrupts in the system. The GSM module has an interrupt, that forces the main program to call the subroutine *INPUT_GSM*, that manages the handling of incoming data from the GSM module. In the same manner, the keypad has an interrupt, that forces the main program to call the subroutine *AUTO_KEY*. This subroutine manages the incoming information, received through the keypad. The reset button does not have an interrupt in the common sense. It merely forces the M68k to reset and start at a specific address.

5.5.2 Program flow

In figure 5.8 the program flow is illustrated. The figure illustrates the flow of the main program and the different system tasks e.g. the subroutines. Not all of the subroutines are triggered by interrupts, actually most of the subroutines are running continuously, without interrupts. This functionality is illustrated by the loop, that points to the different tasks. Some of the subroutines are biased using counters, that hinder the routines from being executed every time, they are called from the main program, *NORMAL*. All of the subprograms contain subroutines, though some are more complex that others. This section describes the functionality and external interface of the subprograms, called from *NORMAL*.

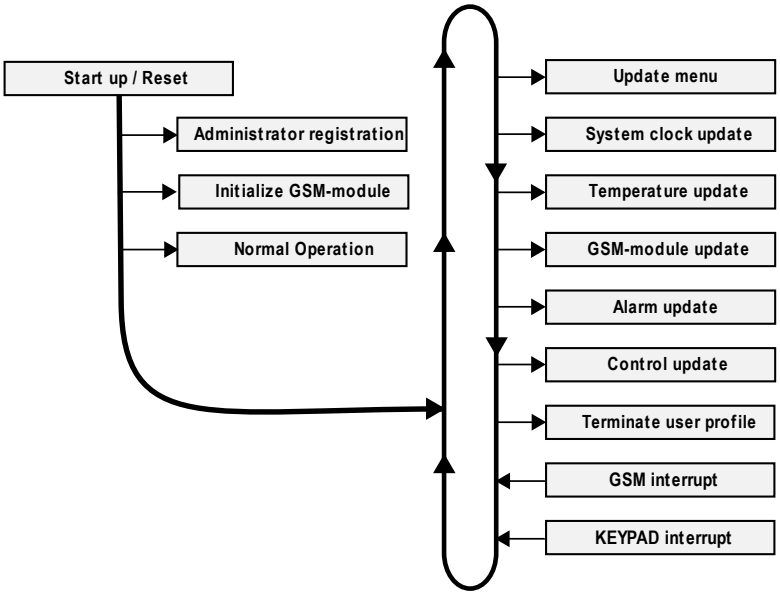

Figure 5.8: Illustration of the program flow.

Start up / Reset

The start up / Reset function is not actually a program in the common sense. It is only listed to indicate the start position of the system initialization.

Initialize LCD

Before the administrator can enter the administrator information, the LCD must be initialized. This is accomplished by configuring the LCD Entry mode and Function set.

Administrator registration

In the actual system program the "Administrator registration" i.e. the entering of the administrator information, is managed by the subroutine *MENUMAIN*. The routine is the standard routine used, when navigating the menus in the local user interface. The routine runs a series of menus, that allow the administrator to enter the administrator phone number, the GSM PIN code, time and date.

Initialize GSM-module

The initialization of the GSM module is managed by the routine *INIT_GSM*. This routine sets up the GSM module to handle text messages used in the system, by sending a series of AT commands to the module.

Normal Operation

The phrase "Normal Operation" refers to the program loop, that the system software runs continuously. The routine *NORMAL* calls the individual subroutines, illustrated in figure 5.8. All of the tasks represent subroutines, that manage a specific part of the system functionality.

Update menu

The update menu actually refers to the routine *MENUMAIN*, that manages the control of and the printing of menus on the LCD. Whenever the function is run, the LCD is updated. When the local user interface is not used, the routine manages the printing of the system status menu on the LCD. Whenever the local interface is used, it manages the printing and the jumping between the individual menus.

System Clock update

The clock is used by the system in relation to keeping track of when a user profile is to be terminated. Since the system does not feature an embedded clock routine, the clock in the GSM module is used. This however, means that the system clock must be updated from the GSM modules clock. This function is managed by the routine *CLOCK_UP*, that updates the system time.

Temperature update

In order to update the current temperature, the routine *ADC* is run. This routine updates the temperature by accessing the ADC, that is connected to the temperature sensor.

GSM module update

Since the system is dependent of the functionality of the GSM module, the status menu features an indication of whether the GSM module is connected to the network or not. In order to update the state of this connection, the routine *GSM_UP* is executed.

Alarm update

The system is always monitoring the temperature, in order to determine if the alarm temperature is reached. If a temperature outside the range specified by the alarm temperature is measured, the system sends an alarm to the administrator. This functionality is managed by the routine *ALARM_UP*.

Control update

When the system is in control mode, the routine *CONTROL_UP* is run. The routine is a compare type function, that uses the set temperature and the current temperature, to determine whether to turn the mock system on or off.

Terminate user profile

Since the system is based on a single administrator/user profile setup, the user profile must be erased whenever a new user wishes to use the system. To ensure that the system does not send any messages to users, that are no longer residents at the system location, the user profile is automatically terminated after a user specified number of days. This action is managed by the routine *TERMINATE_USER*, that uses the variable "Days active" and the system clock in order to determine when to terminate a user profile.

GSM Interrupt

The GSM_input is the routine responsible of managing the incoming data from the GSM module, When the GSM module sends data the system needs to be ready to receive it, or it will be lost. To disturb the system, as little as possible and still be able to serve the GSM module, the interrupt routine will only move data to the buffer, and count the amount of line feeds that are received.

KEYPAD Interrupt

The KEYPAD interrupt manages the entry from the keypad and stores the entry data. This is the only action performed when the keypad is pressed. The routine is deliberately kept as small and simple as possible, to ensure that the interrupt routine does not become too time-consuming.

5.6 GSM software

In order to use the GSM module in the system, there are some basic functions that needs to be created. These functions will create the basis for how the rest of the software communicates with the GSM module. The main objective is to allow a single function, manage all the functions related to the GSM module. By this definition, the system sets up a series of variables, before sending a command to the GSM module. The function waits for a response and another function handles the response. The actual program code will not be shown in this section, the code in this section is pseudo code, the actual code can be found in appendix H on page 122.

5.6.1 Setting up the ACIA

The ACIA has to be set up in order to interact with the system. The ACIA is configured to send interrupts, when it receives data from the GSM module. Furthermore the ACIA has an internal clock divider that is used to control the transfer and receiving baud rates. The incoming clock frequency is designed for the ACIA to divide by 16, calculated in section 4.5.3 on page 47. Besides the baud rate, the ACIA can send data in several different formates, for the GSM module the default format is 8bit data, 1 stop bit and no parity, and this will be the format used in the designed system. The last configuration is of the ACIA interrupt system, the ACIA is capable of sending interrupts, when it is ready to send and when it has received data. To work with the build up of the rest of the software, the transmitting interrupt are disable

and the receiving interrupt are enabled. As described in section 4.5 on page 45, the ACIA is setup by a control register. The register is placed at address $B00001.

Software functionality

The counter divider is set to 16, by setting bit 0, to 1, and bit 1, to 0. The transfer format is set to 8bit data, 1 stop bit and no parity. Bits 2-4, are set to 1 0 1 for this . Transmit control is disabled, as the system does not use interrupts to transmit data. To disable the transmit interrupt, the bits 5 and 6 are set to 0 0.
The receiver interrupt is set at bit 7. This gives the set up of the control register as 11010101 or 0xD5.

Pseudo code

```
SETUP_ACIA
ACIA_CR = $D5
END
```

5.6.2 GSM module commands

All commands to the GSM module, are bounded to a syntax. All commands is starting with AT, and will be executed with a <CR> (Carrier Return) or 0x0D, according to the datasheet [15], where all features are described in greater detail.

Initializing the GSM module

The GSM module has to be initialized. To do this, the GSM module, needs several commands transmitted in a specific order. The used commands are in table 5.3. In the table they are listed in the order, they are sent to the module.

COMMAND	DESCRIPTION	ARGUMENT
ATE	Turn off/on echo from the module	0
AT+CPIN	Registers the pin code for the SIM card	"XXXX"
AT+CNMI	Sets the GSM module to send an interrupt signal when a SMS is received	1,2,0,0,1
AT+CMGF	Set the message mode for the module	1
AT+CCLK	Controls the internal clock in the module	YY/MM/DD,HH:MM:SS

Table 5.3: AT-commands for initializing the GSM module.

Handling SMS messages

The primary function of the GSM module will be to manage text messages from administrator/user. The commands used to handle text messages are shown in table 5.4.

COMMAND	DESCRIPTION	ARGUMENT
AT+CMGS	Sending SMS messages	Phone number of receiver
AT+CMGR	Reading SMS messages	index for SIM storage
AT+CMGD	Deleting SMS messages	index for SIM storage

Table 5.4: AT-commands for handling text messages.

Misc AT commands

In addition to these commands in tables 5.3 and 5.4, there are a few commands that do not fit into thees categories. These are the commands for receiving information from the GSM module, about the GSM network, the time from the internal clock and the command for hanging up on call.

COMMAND	DESCRIPTION	ARGUMENT
AT+CSQ	Get the signal strength for GSM network	None
AT+CCLK	Getting the time form the module	None
AT+CHUP	Hanging up a call	None

Table 5.5: Misc AT-commands.

In order to send the commands to the GSM module, the software writes to the ACIA data register on the address $B00003. The ACIA has an 8 bit data bus, this means that all commands and text strings, have to be sent one character at a time. All commands and text strings are preprogrammed in the data area of the memory, as described in section 5.3.2 on page 71.

Code function

The function will determine if the ACIA is ready to transmit, before the system attempts to send data to the ACIA. To make the function usable in all situation, it is designed in such a way, that it will transmit the data from a pointer. This pointer is set to point at the first character/byte of a string, before the function is executed. As the function is designed to access data from a pointer, and then move the pointer one byte at a time, until it reaches the end of the string. The function will recognize the end of the string, by a 0x00 byte in the last space of the array. Along with the pointer all variable arguments for the commands, will be set too.

Pseudo code

```
        LOAD POINTER TO STRING
        SET ARGUMENTS
START   SEND_ST_GSM
L1        IF (ACIA != READY)
            RETURN TO L1
          IF (CHAR != $00)
            ACIA_DR = CHAR
            ADD 1 TO POINTER
            RETURN TO L1
END       END
       SET POINTER TO ARGUMENTS
       SEND_ST_GSM
```

5.6.3 Response from GSM module

Reading data sent by the GSM module, is accomplished by running an exception, that is similar to a subroutine. When an interrupt is sent from the ACIA, this exception will move the data from the ACIA onto the system buffer, as described in section 5.3.2 on page 72. To indicate, when a command is completely stored in the buffer, the function will count line feeds, 0x0A. The reason for line feeds to be counted, is that the syntax for GSM response, always contains the same amount of line feeds. The commands, in most cases contain two line feeds in the response. However, when reading a text message, there will be five line feeds, a single line feed, when the GSM module is ready to receive a SMS messages, and four line feeds after the CCLK and CSQ commands. This is implemented by designing an additional four smaller functions, that will wait for the right amount of line feeds, when expecting a response from the GSM module.

Software functionality

The function will, when triggered, move a byte from the ACIA to the buffer, where the data-in pointer, points to. For each byte of data, the function will compare it to 0x0A, to count the line feeds, and add one to a variable for every line feed. As the function is interrupt controlled, it can be executed at any time, which means that special care must be taken, when handling the internal registers of the processor. The internal registers, must be the same before and after the function is run.

Pseudo code

```
START   INPUT_GSM
          MOVE USED REGISTERS TO STACK
          MOVE ACIA DATA TO A REGISTER
          IF ( DATA == $0A)
           ADD 1 TO LF COUNTER
          MOVE DATA TO BUFFER IN POINTER
          ADD 1 TO DATA IN POINTER
          MOVE ORIGINAL REGISTER BACK FROM STACK
END       END
```

5.6.4 Recognizing command lines

In order for the system to handle commands and recognize text strings in text messages, it needs to be able to recognize character strings, and link a function to it. To do this, all accepted commands and responses from the GSM module is saved in the system memory as character strings. The character strings and subroutines are linked together in a linked list, described in section 5.3.2 on page 72. By using a linked list, the function only needs to know the address of the first element, and with NULL placed in the last elements pointer. The function will run a linear search in the elements. This type of search has a high complexity (n^2), but as there are relatively few possibilities in the system, there is no need for a more efficient algorithm.

Code function

The function will compare the first character in the buffer with the first element and continue on to the next. If presented with two characters that do not match, the code will start over with the next element. If

an entire string is matched, the address for the elements function, will be saved in a register so it can be used afterwards. If the string is not recognized, the code runs an error function.

Pseudo code

```
START   CMP_ANS
          SET ELEMENT POINTER TO FIRST ELEMENT
L1        SET POINTER TO SAVED DATA
          SET  READ POINTER TO START OF BUFFER
L2        IF (SAVED DATA POINTER == $00)
            EXECUTE SUBROUTINE FROM ELEMENT
END         END
          IF (READ POINTER ADDRESS == BUFFER END POINTER ADDRESS)
            EXECUTE ERROR SUBROUTINE
END         END
          IF (READ POINTER == SAVED DATA POINTER)
            ADD 1 TO SAVED DATA POINTER
            ADD 1 TO READ POINTER
            JUMP TO L2
          ELSE
            SET ELEMENT POINTER TO NEXT ELEMENT
          IF (ELEMENT POINTER == NULL)
            EXECUTE ERROR SUBROUTINE
END         END
          ELSE
            JUMP TO L1
```

5.6.5 Accepted strings and actions

Every time the *CMP_ANS* function successfully identifies a string from the GSM module, it needs to perform the correct action. All accepted strings are stored in a linked list, described in section 5.3.2 on page 72, as the same compare function is used for all strings, this includes both responses from the GSM module and commands for the administrator/user, written as text messages.

Responses from the GSM module

All responses from the GSM module starts with the characters <CR><LF> (0x0D 0x0A), these are not shown in table 5.6, where all used responses, that trigger a function in the software, are listed.

Activation string	Description	Action
'OK'	Command accepted	Set a bit flag
'+CME ERROR: incorrect password'	Wrong pin code	Set a bit flag
'+CME ERROR: invalid index'	Invalid argument for command	Set a bit flag
'+CMTI: "SM",'	SMS received	Receiving SMS messages*
'+CMGS:'	SMS sent	Set a bit flag
'+CSQ: '	GSM network status	Updates the GSM status
'+CCLK: '	Clock	Updates the systems time
'> '	Ready to write SMS message	Transmits a SMS messages*
'RING'	GSM module is receiving a call	Transmits the hang up command

Table 5.6: Known responses from GSM module, *Described later in this section.

All the responses, that trigger the setting of a bit flag are used in the larger functions to control that the GSM module has done as commanded, etch responds set a unused bit in a internal register, when doing it like this can all critical commands be controlled just by looking at on bite.

5.6.6 Text message commands

The commands are given in the manual in section 3.7 on page 28. All the commands trigger a function, similar to the GSM responses and are automatically executed by the *CMP_ANS* function. To get the *CMP_ANS* function to work, on the SMS messages, the buffer start pointer, that are used in the function to show the beginning of a string, is temporarily moved to the beginning of the SMS message. This is done to make the CMP_ANS function look at the SMS, instead of the whole response line from the module.

Activation string	Description
'alarm off'	Disables the alarm function
'stop'	Clear control status bit
'status'	Sends a Status SMS*
'settemp'	Saves the arguments
'setalarm'	Saves the arguments

Table 5.7: Accepted orders in SMS messages, *Described later in this section.

All text message triggered commands, sends a confirmation back to the administrator/user. All commands are scanned for conflicts in variables and for syntax errors. The scan will trigger the sending of an error message, if any abnormalities are detected.

5.6.7 Sending text messages

To send text messages, the system needs to send a command, a phone number and the message to the GSM module. To minimize the function code, the function will use the earlier described functions.

Code function

To send a text message, the command AT+CMGS, is sent to the GSM module. This is done by setting a pointer to the command in the data area, and running the *SEND_ST_GSM* subroutine. The same approach is used with the number. To get the command syntax right, it is needed to send a <CR> character after the number. After this, the *CMP_ANS* function will be triggered, to make sure the GSM module is ready for the message, before transmitting it.

Pseudo code

```
START   SEN_SMS
          SET FUNCTION POINTER TO COMMAND
          SEND_ST_GSM
          COPY NUMBER POINTER TO FUNCTION POINTER
          SEND_ST_GSM
          SET FUNCTION POINTER TO COMMAND END CHAR
          SEND_ST_GSM
          CMP_ANS
END       END
```

5.6.8 Transmitting a text messages

When the *CMP_ANS* function identify the '> ' response from the GSM module, the system starts to transmit a message.

Software functionality

Before the *SEND_SMS* function is triggered, the system will have given a number for the message. Each number corresponds to a subroutine, that uses the *SEND_ST_GSM* function to send a series of strings and/or variables to the GSM module, that combined makes up the SMS message.

Pseudo code

```
START   TRANS_SMS
          EXECUTE SMS FUNCTION MSG_NR
END       END
```

5.6.9 Receiving text messages

When the *CMP_ANS* function identifies, that the GSM module has received a new message, it is set up to send a command string, with information on the storage location of the message.

Software functionality

In the main loop, there is a function that looks at the number of line feeds, received at a given moment. If anything is received, the system will halt and wait until two line feeds are registered. When the line feeds are registered, the response for the GSM module will be placed in the buffer, and the *CMP_ANS* function will be executed. Subsequently, the appropriate command string, corresponding with a text message is

received, the function finds the index number for the message, stored on the SIM card. By sending the AT+CMGR command and the index number, the GSM module will send the message to the system buffer. Here the system will wait for five line feeds. When the entire message is received, a function will find the phone number and save it in a temporary variable. After the number is saved, the number is compared to the stored numbers for user and administrator. The function will set a different bit flag, depending on whether the message comes from a user, administrator or an unknown user. if the used phone number is not authorized, the system will respond with an error message. If the system recognizes the number, the system will temporarily move the start pointer of the buffer to the second line feed, which is where the messages begins. Then *CMP_ANS* is run, to determine which subroutine to execute.

Pseudo code

```
START   READ_SMS
          SET POINTER TO READ COMMAND
          SEND_ST_GSM
          SET POINTER TO MSG_NR
          SEND_ST_GSM
          FIND_NR
          CMP_NR
          SET POINTER TO SMS START
          CMP_ANS
END       END
```

5.6.10 Deleting text messages

The system will delete messages, when they have been handled. This is done to prevent the SIM card from filling up.

Software functionality

The function will send the AT+CMGD, along with the storage index of the message, to the GSM module. After the command is sent, the system will wait for two line feeds to be received and execute the *CMP_ANS* function to confirm that the system responded with OK. OK confirms that the message was deleted. If the GSM module sends any other response, the system will try again, until the proper response is received.

Pseudo code

```
START   READ_SMS
L1        SET POINTER TO DELET COMMAND
          SEND_ST_GSM
          CMP_ANS
          IF(OK BIT != 1)
           JUMP TO L1
END       END
```

5.7 Local interface software

This section describes the design of the software, related to the local user interface and contains a description of the general program design, based on the functionality of the related devices. It also deals with the process design of the individual modules.
The interface consists of two parts, the LCD and the keypad. The LCD is a part of the local user interface and is placed on the address bus. Its write address is placed on address $900003 and the control address is located on $900001. Whenever commands or data are to be sent to the LCD, these two addresses are used.
The LCD software uses a series of registers and local variables. The global variables are listed in table 5.1 on page 71, and the local variables are listed, as they are used in the program code. The keypad is the other part of the local user interface and it allows the administrator/user to enter data to the system. The keypad is placed on the data bus, and the control address is placed on address $A00001. The keypad has interrupt level 5 and is therefore the highest masked interrupt in the system.

5.7.1 Program and Menu design

The function of the menus is to visually communicate information to the administrator/user. This is done by displaying a series of menus containing information, related to the different features of the system. The menu path diagram, that illustrates the way the different menus are connected, can be seen in figure 5.9. The paths and numbers indicate the buttons, that must be pressed, to reach a specific menu.
All of the menus consists of individual subroutines, that follow the same overall program structure, but vary in complexity. The administrator/user navigates the menus using the keypad. Note, that the administrator is forced to enter the administrator information upon start up/reset.
The administrator information is used during the initialization of the software, that takes place after the administrator has finished entering the information.
Provided that the system is not turned off or reset, the normal usage of the local user interface is always initiated by pressing any key on the keypad and navigating the menus, beginning from the "Main menu".
Should the administrator need to reset the system, i.e. if the system fails, he/she must press the reset button and is again required to enter the administrator information. The writing of the menus on the LCD, is managed by the subroutine *MENUMAIN*. *MENUMAIN* is incorporated in the main program *NORMAL* in a way, that ensures that the system variables are updated and the other subroutines are periodically executed, while the local user interface is used. This is accomplished by accessing each menu using a branch function, that uses an argument that is linked to a specific menu. Whenever a menu is executed, the last command sets the menu variable in accordance with the next menu. Because *MENUMAIN* is part of *NORMAL*, the rest of the main program is run, before the next menu is printed. This could pose a timing problem, since the rest of the main program is time-consuming. This, however, is not expected to pose any problems, since the main program is limited in scale and therefore is quite rapidly executed. The actual time consumption will be evaluated at the completion of the system software. The flow of the subroutine *MENUMAIN* in relation to the main program is illustrated in the pseudo code below.

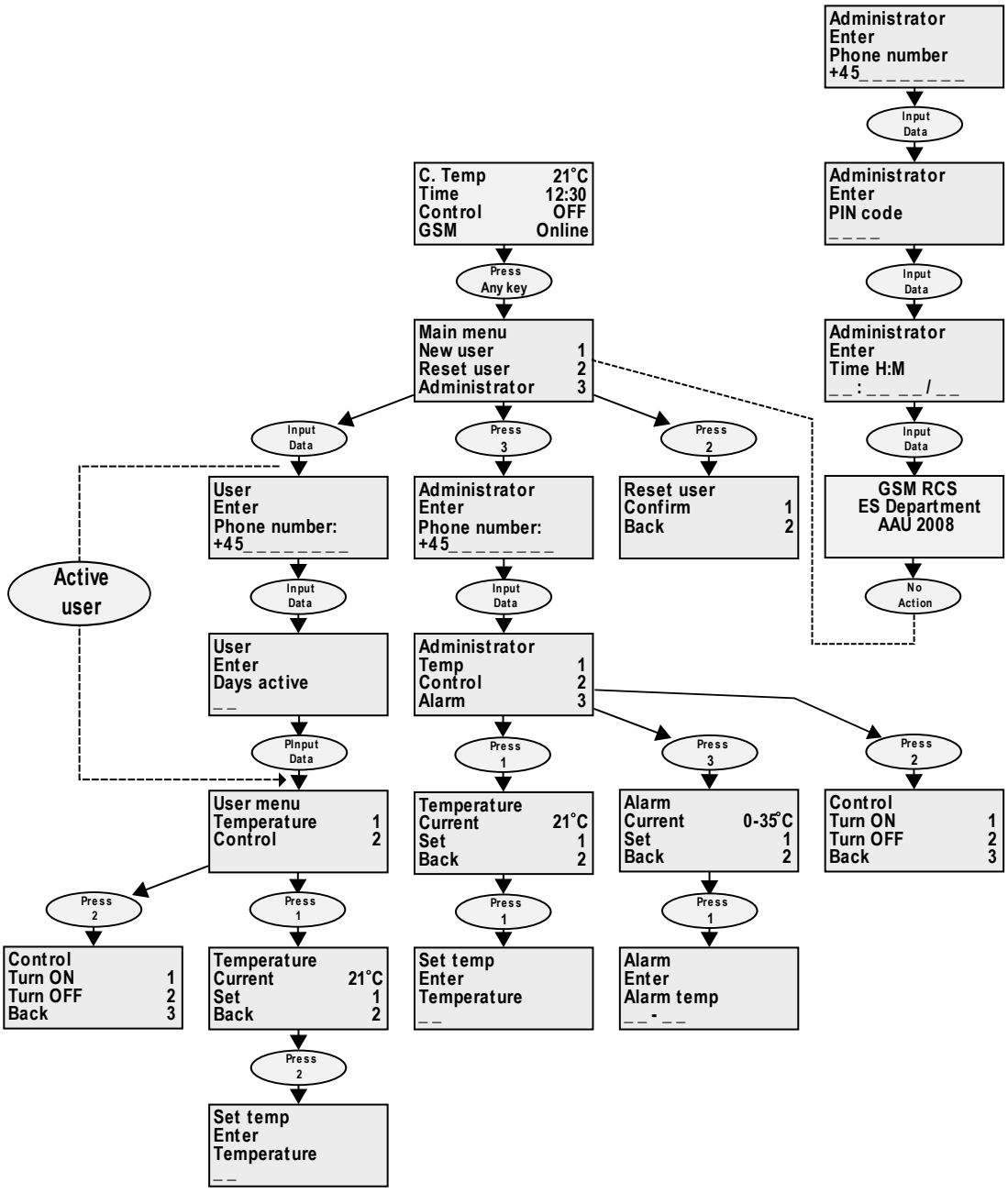

Figure 5.9: Illustration of the menu system that makes up the local user interface.

Pseudo code

```
MENUMAIN              LOAD USER SELECTED MENU FROM D3
                      GET USER INPUT FROM KEYPAD
                      SET D3 TO NEXT USER SELECTED MENU
                      RETURN TO MAIN LOOP
```

The menus used in *MENUMAIN* can be divided into four types, that are defined by the amount of entries they operate with. The four types of menus are listed below.

- JUMP MENU
  The menu is used to navigate the menu system. It only uses single character numeric entry to do so.
- (2) NUMERIC MENU
  The menu is used when a two-character entry is made. This menu type is for example used, when entering the set temperature or the number of days a profile is to be active.
- (4) NUMERIC MENU
  The menu is used when a four-character entry is made. This menu type is used when entering the alarm temperature.
- (8) NUMERIC MENU
  The menu is used when an eight-character entry is made. This menu type is used when entering phone numbers.

These four menu types make a total of 21 menus, that are accessed by *MENUMAIN*. They all follow the same basic structure, although some of the menus are more complex than others. The basic structure of the menu functions are illustrated below.

Pseudo code

```
MENU_EXAMPLE          EXECUTE "RESET LCD"
                      EXECUTE "SET CURSOR MODE"
                      READ MENU FROM D3
                      EXECUTE PRINT MENU ON DISPLAY
                      UPDATE THE VARIABLES IN THE MENU
                      PRINT VARIABLES TO DISPLAY
                      GET USER INPUT FROM KEYPAD
                        IF INPUT = 1
                          SET D3=MENU 1
                        ELSE IF INPUT = 2
                          SET D3 = MENU 2
                        ELSE IF INPUT = 3
                          SET D3 = MENU 3
                      ELSE GO TO END
END                   RETURN TO MAIN LOOP
```

All the menu subroutines begin by calling the function *RESET_LCD*, that clears the display. After the reset function has been executed the cursor mode is set. In the jump menus, the cursor is set not to blink and in the entry menus the cursor is blinking. The next action is the actual printing of the menus, that is accomplished be sending a series of ASCII characters to the LCD. The text lines, used in the menus, are stored in the RAM memory beginning from address $45000. They are stored as text strings, that end with 0x00. This eases the printing of the menus as the 0x00, is used as a stop bit. The actual printing function *PRINT_LINE* is described later on.
If the menu contains variables, these are updated. This is accomplished by printing the variables on their respective location in the menu, using jump commands to move the cursor to the right place. The next action is that the routine waits for the administrator/user to make an entry. If no entry is made within 20 seconds the menu returns to the main program and sets the menu variable so the next menu printed on the LCD is the status menu. The specific menu in question, determines the processing of the entry from the

keypad. The input from the keypad is obtained, using the subroutine *KEYPAD_C*, that identifies the entry using a series of compare functions. Before returning, the entry is converted to a number between 0 and 9, or *. This conversion is necessary since the encoder, used in relation to the keypad, encodes the pressed buttons. The actual entry is managed by the interrupt function *AUTOKEY*, that saves the converted entry from the keypad as the variable entry, whenever a button is pressed.
Before returning to *MENUMAIN*, the menu argument is updated in accordance with the choices displayed in the menu and the entry.

5.7.2 Program modules

In this section the functionality of the program modules, used in *MENUMAIN* are described. The code related to the LCD is located in appendix I.1 on page 136.

PRINT_LINE

PRINT_LINE is the standard module for printing characters to the LCD. In order to use the module, it is necessary to set the pointer, A6, to point at the beginning of a text string. Each character of the string is then printed to the LCD one by one, until the end of the string. This is accomplished by incrementing the pointer one bit at a time and printing the content of the address to the LCD. To ensure that the LCD can keep up with the processor, the LCD busy flag is read, before a character is sent to the display.
In order to determine if the end of the string is reached, all the text strings are ended with 0x00. Before the character is printed to the LCD, it is compared to 0x00 and the module terminates if the character is in fact NULL.

INIT_LCD

INIT_LCD covers the initialization of the LCD. The module sends a series of commands to the LCD, using the address bus. The LCD is reset twice, and Entry mode is initialized along with Function set. Entry mode is set to auto increment right. Function sets 8 bit data length, the number of lines is set to four and the font is set to 5x7 pixels.
To ensure that the commands are accepted by the LCD, every command is followed by a wait function *WAIT*, that ensures that the commands are separated by at least 15ms.

Location modules

Many of the menus, use several of the LCD lines, which call for functions that can control the location of the cursor and move it to a specific location. The modules *SECOND*, *THIRD* and *FOURTH*, move the cursor from its current position to the first character on the second, third and fourth line respectively. The actual jump is accomplished by sending a location command to the LCD, followed by the wait function, *WAIT*.

RESET_LCD

The module *RESET_LCD* is used to reset the LCD, whenever a new menu is to be displayed on the LCD. The module sends the reset command, 0x01, to the LCD, followed by the wait function, *WAIT*.

ONOFF

The module *ONOFF* is used to determined whether the system is in control mode or in monitoring mode. This module is used in relation with *PRINT_LINE*, when printing the state of the system in the status menu. It checks the first bit in the register D4, that holds the status bit related to the state of the system. If the bit is equal to 1, the A6 is set to point to the text string 'ON'. If the bit is equal to 0, the pointer is set to point at the text string 'OFF'.

GSM_ONOFF

The module *GSM_ONOFF* is used to determine, whether the system has access to the GSM network or not. This module is used in relation with *PRINT_LINE*, when printing the state of the system in the status menu. It checks the second bit in the register D4, that holds the status bit related to the state of the GSM module. If the bit is equal to 1, the A6 is set to point to the text string 'Online'. If the bit is equal to 0, the pointer is set to point at the text string 'Offline'.

KEYPAD_C

KEYPAD_C is the module, that converts the entry from the keypad to a number between 0 and 9 or *. The module is necessary because the encoder, used to control the keypad, does not map the pressed keys into ASCII characters. The function consists of 11 compare loops, that determines the entry and move the resulting ASCII character into register D5. This entry is later used to navigate the menus and in the process of entering administrator/user information.

5.7.3 Summary

In this section the design of the software, related to the local interface has been described. The basic structure of the subroutine *MENUMAIN* and its sub process have been described.

5.8 ADC software

To make sure that the system can read the temperature, a conversion of the binary value, which the ADC outputs, is needed. In this section the software of the Temperature sensor and ADC is described.
When the ADC puts its binary value on address 0xC00001, the software has to convert the binary value into decimal representation, of the temperature. This requires some extended math, but due to limitations in the processor, some of the calculations are quite complicated. Therefore the calculation must be reduced into fractions and in some cases simplified.

5.8.1 Structure

The ADC software is divided into two parts. The actual main code and the math calculations. In appendix J the software for the temperature sensor can be seen.

5.8.2 Temperature calculation

The ADC converts the voltage of the temperature sensor into a binary value, which the M68k interprets as a decimal value. This value is the number of bit steps from the lower reference, and by calculations, it can be converted into a temperature in $^{\circ}C$. This temperature is used several places in the system software, i.e. in the control function *CONTROL_UP*.

Temperature calculations

The calculation of the temperature is based on a simplified calculation, that originates from eq. 5.1.

$$Temp = \frac{(x \cdot LSB) + V_{Ref-}}{V_{REF}} \cdot T_{REF} \tag{5.1}$$

In the calculation. x is the bit size from the ADC. From calculations 4.19 on page 50, the voltage per bit (LSB) is determined, to $4.688 \cdot 10^{-3}V$.
V_{Ref-} is determined in equation 4.16 on page 50 to $2.53V$.
The V_{REF} and T_{REF} are the referens pointe used in the equation, V_{REF} is set to $2.982V$, corresponding to the T_{REF} which is set to $25 + 273.15^{\circ}K$

$$Temp = \frac{x \cdot 4.688 \cdot 10^{-3}V + 2.53V}{2.982V} \cdot (25 + 273.15^{\circ}K) \tag{5.2}$$

$$Temp = \left(x \cdot 4.688 \cdot 10^{-3}V + 2.53V\right) \cdot \frac{298.15^{\circ}K}{2.982V} \tag{5.3}$$

$$Temp = x \cdot 4.688 \cdot 10^{-3}V \cdot \frac{298.15^{\circ}K}{2.982V} + 2.53V \cdot \frac{298.15^{\circ}K}{2.982V} \tag{5.4}$$

$$Temp = x \cdot 468.75 \cdot 10^{-3} \cdot^{\circ} K + 253.15^{\circ}K \tag{5.5}$$

$$Temp = 468.75 \cdot 10^{-3} \cdot^{\circ} K \cdot \left(x + \frac{253.15^{\circ}K}{468.75 \cdot 10^{-3} \cdot^{\circ} K}\right) \tag{5.6}$$

$$Temp = 468.75 \cdot 10^{-3} \cdot^{\circ} K \cdot (x + 540.05) \tag{5.7}$$

The processor does not support floats and therefore, values are rounded off and/or substituted by fractions, to make the calculation possible.

$$540.05 \approx 540 \tag{5.8}$$

$$468.75m \approx \frac{15}{32} \tag{5.9}$$

The function will add 540 to the value of the ADC, multiply it by 15 and divide it by 32. After the calculation, the function will have the temperature in $^{\circ}K$. To convert it into $^{\circ}C$, the function will subtract 273, since it is the approximate difference between Kelvin and Celsius. While converting the value, the function needs to determine, if the temperature is positive or negative. The right signs are set in the final value, along with the temperature, converted into ASCII characters.

Pseudo code

```
START   GET TEMPERATURE FROM D0
        CONVERT BITS TO KELVIN
        PUT KELVIN ONTO D0
          IF (D0 <= 273)
            D0 = 273-D0
            D2 = '-'
          ELSE
            D0 = D0-273
            TEMP_C_D = D0
            D2 = '+'
        TO_10()
```

The function *TO_10()* has the purpose of integer division and outputting the temperature as ASCII characters. First, D1 is divided by 10 to get the 10s. D2 is rotated as a longword to the left, and D1 is afterwards put on D2. The left over value in D1 are the amounts of 1s, D2 are rotated again and D1 is moved to D2. Finally, the code determines whether there should be .0 or .5 after the integer temperature. Adding $303030 to D2, converts the number into ASCII characters. At last the temperature is moved to D7.
An example of this could be the $+36.0°C$. In D7 it will have the format $2B333630. $2B is the ASCII code for '+', and $30 - $39 is the code for numbers 0-9. With a negative temperature such as $-6.5°C$ will be converted to $2D303635 in ASCII.

5.8.3 Summary

In this section, the calculations in relation to the temperature sensor, has been described. The calculation implementation in the software has also been addressed.

5.9 Alarm update

To enable the alarming of the administrator, it is necessary to design a function, that monitors the current temperature. The current temperature is compared to the alarm temperatures, in order to decide whether an alarm should be dispatched.

5.9.1 Detecting alarm temperatures

The function requires a predefined set of alarm temperatures, set by the administrator. If these, are not set by the administrator, the system uses the default values. These values, a maximum allowed temperature and a minimum temperature, must be compared with the current temperature, to determine if an alarm must be dispatched.

Code function

The values stored in the alarm temperature and set temperatures are loaded by the function, followed by the current temperature. When this is done, the current temperature is compared to the maximum temperature. If the current temperature equals or exceeds this value, and alarm is dispatched. If no alarm is detected, the current temperature is compared to the minimum temperature. If the current temperature is below or equal the minimum temperature, an alarm will be dispatched.

Pseudo code

```
                    LOAD MIN AND MAX TEMPERATURES
START               COMPARE MIN_TEMP WITH CUR_TEMP
                      IF (CUR_TEMP >= MAX_TEMP)
                        GO TO ALARM
                      ELSE
                        IF (CUR_TEMP =< MIN_TEMP)
                          GO TO ALARM
                        ELSE
                          GO TO END
ALARM               SEND ALARM SMS
END                 END
```

5.10 Terminate user

The terminate user function is used to terminate the user profile from the system, when the days active has elapsed. The function must be able to detect, when a day has passed, even if the user is using the menus at the time, the clock indicates a day has passed. The function will detect a day has gone at midnight, but the user should not be terminated, until 12:00. The function is only executed if an user is active. It is noted that the user profile can always be terminated from the main menu on the local interface.

5.10.1 Detection of days elapsed

Every time the function is executed, it will check the system time, to see if it is past midnight. If the clock has passed midnight, since the last check, the function will update how many days has gone by.
If the user is using the menu for a few minutes, when the clock passes midnight, the function will not detect that it has been midnight (00:00). To compensate for this problem, a buffer zone is added, so that the function will determine, if it is past midnight, up to 30 minutes after midnight (00:00 to 00:30).
The opposite problem may also occur, especially with the buffer zone implemented, since the system may now think a day has passed every single time the function is run, in this 30 minutes period. The solution to this is a piece of code, that will let the system know, if the current day has already been counted.

5.10.2 Terminating the user

When the day on which the user must be terminated has dawned, the system will start running a new piece of code. This code is able to terminate the user profile from the system, but will first check if the time is 12:00 or later.

Pseudo code

```
                    CHECK FOR ACTIVE USER
                      IF NONE
                        GO TO END
                      ELSE CONTINUE
START               IF (00:00 =< TIME < 00:30)
                      IF (DAY\_COUNTED = NO)
                        DAYS\_GONE + 1
```

```
                        DAY\_COUNTED = YES
                      ELSE CONTINUE
                    IF (DAYS\_ACTIVE =< DAYS\_GONE)
                      GO TO TERM
                    ELSE GO TO END
TERM                  IF (TIME >= 12:00)
                      DEACTIVATE USER
                    ELSE GO TO END
END                 END
```

5.11 Control Update

The control update function is used to determine if the system is in control mode. If the system is in control mode, the function must determine whether the mock system should be turned on or off (simulating turning a heater on or off).

5.11.1 Adjusting Temperatures

The temperature control is in a many ways, similar to the alarm update. The temperature control compares the current temperature to the set temperature.
Since temperatures tend to fluctuate and the fact that the temperature sensor also might flicker between different temperatures, the temperature control needs to have a "dead zone". The "dead zone" is a small temperature range, around the set temperature, where the function ignores if the current temperature is not the exact set temperature. This is a necessity, since the administrator/user has no interest, in having a heater turning on and off several times per minute, nor will the heater be very effective operating in this manner. Therefore the "dead zone" is designed so the function allows the temperature to rise 1^oC over the set temperature and then allow it to fall 1^oC below the set temperature, before reactivating the mock system.

Software function

The first action, is to determine whether the temperature control is enabled or not. If not, the function is terminated immediately. If temperature control is enabled, the function loads the temperature, the system tries to obtain. If the current temperature is higher than or equal to SET_TEMP+1, the mock system is turned off. If this is not the case, the function determines whether the current temperature is lower than or equal to SET_TEMP-1. If the temperature is lower that the set temperature, the mock system will turn on. In the case that none of these comparisons are fulfilled, the control unit will stay in its current state and the function will terminate.

Pseudo code

```
                IF TEMPERATURE CONTROL = OFF
                GO TO END
                ELSE CONTINUE
                  LOAD SET\_TEMP
START                 COMPARE (SET\_TEMP+1) WITH CUR\_TEMP
                IF (CUR\_TEMP >= SET\_TEMP)
                  TURN OFF CONTROL UNIT
```

```
                COMPARE (SET\_TEMP-1) WITH CUR\_TEMP
                  IF (CUR\_TEMP =< SET\_TEMP)
                  TURN ON CONTROL UNIT
                  ELSE
END                 END
```

5.12 Implementation

In this chapter the implementation of the software is described. The chapter also addresses the debugging and initial testing of the system software.

5.12.1 Software debugging

During the design and testing of the system software, the actual software is run using the TS2 monitor, and not the minimal system ROM. Using the TS2 monitor, gives the programmer many possibilities in relation to debugging the software. The actual features of the monitor, are described in section 5.4 on page 73.

The main focus of the system software, is on the locale user interface and the GSM communication. This is done, since these two subjects are the most comprehensive.
The main program *NORMAL*, is written as the first of the programs. This is done in order to use the function as a platform, that can be expanded upon, as additional subroutines are developed and implemented. The program, which only consists of a series of function calls, is used to test the different subroutines, as they are implemented. This modularization of the software is possible since, the software is designed in accordance with the SPU model.

As the different subroutines are implemented, they are debugged using the TS2 monitor. This is done by uploading the developed software to the system RAM and running it, through the TS2 monitor. The initialization subroutine, *INIT_LCD*, is the first subroutine implemented. The reason for this is that the routine enables the LCD to be used during the debugging and testing of the software. One way the LCD is used in the debugging, is to write test text to the LCD. This allow the programmer to monitor the flow and the execution of the program in a visual manner.
The second routine, that is implemented, is the natural expansion of the use of the LCD. The subroutine, *MENUMAIN*, allow the programmer to monitor the state of variables and the system operating mode, through the system menus. The implementation of the subroutine ‚*MENUMAIN*, is also done in steps, as the different menus follow the same basic structure. Again, this modular structure allow the programmer to commence the testing of other subroutines, before the entire, *MENUMAIN*, routine has been implemented and tested.
When all subroutines has been implemented in the main program, the software can be tested according to the demands specified in the performance specification, 3.5 on page 23.

5.12.2 Permanent Implementation

Upon completion, the system software could be burned to the ROM, instead of being run form the RAM through the TS2 monitor. This however calls for a close inspection of the software structure, and minor adjustments to the code. The adjustments are necessary, because the interrupt vector initialization and the location of the program code have to be reconsidered. In this project, the actual implementation of the software into the ROM is omitted, since the burning of the ROM only limits the use of the system, as monitoring and developing becomes more difficult.

The use of the monitor as a means of running the software is preferred, since the control of the systems actions and the corrections of any possible errors, are much easier, when using the monitor. Secondly, the actual burning of the ROM proves nothing more, than the system is operational and in accordance with the performance specification. This is demonstrated in the acceptance test, in chapter 6 on the next page.

5.12.3 Summary

In this section, the implementation and debugging of the software has been addressed. The software has been debugged, using the debugging tool TS2 monitor and the software has been loosely tested, according to the demands in the performance specification. The actual testing of the assembled system and software is addressed in the acceptance test in chapter 6. The finished system, is not implemented in the ROM, but is run using the TS2 monitor. This is done, since the group sees few advantages and challenges in running the software from the ROM, instead of using the monitor.

Chapter 6

Accept tests

The acceptance test is a tool used to investigate whether the designed product meets the demands stated in the performance specification. Furthermore the tests are a vital part of the SPU model, which this report is build upon. The demands for the system can be seen in section 3.5.1 on page 23. The test paragraphs, see 3.5 on page 23, has been compiled from the functionality driven demands. These test paragraphs are used to execute tests on the system. The test executions and circumstances are described in three different test journals and one questionnaire.

The test results are stated in this chapter, but the full details on the tests, are stated in the test journals in the appendices. The different test paragraphs are stated with their corresponding results.

6.1 I/O device test results

The test journal for the I/O devices can be found in appendices B on page 108.

6.1.1 § 3.6.1.1 Temperature measurement I

The test seeks to determine, whether the temperature sensor can measure the temperature, with an accuracy of at least 0.5°C.

Test results

Below are the results from the test. In graph 6.1 the collected measurements can be viewed. In graph 6.2 the average between the three thermometers can be seen, together with the sensor measurement. The actual measurements are stated in table 6.2.
It is noted, that the three thermometers measurements are quite far from each other. Actually, the temperature sensor is closest to the average, between the three thermometer measurements. The biggest aberration is on 0.8°C and the smallest is 0.03°. The aberrations between the average and the temperature sensor can be seen in table 6.1.

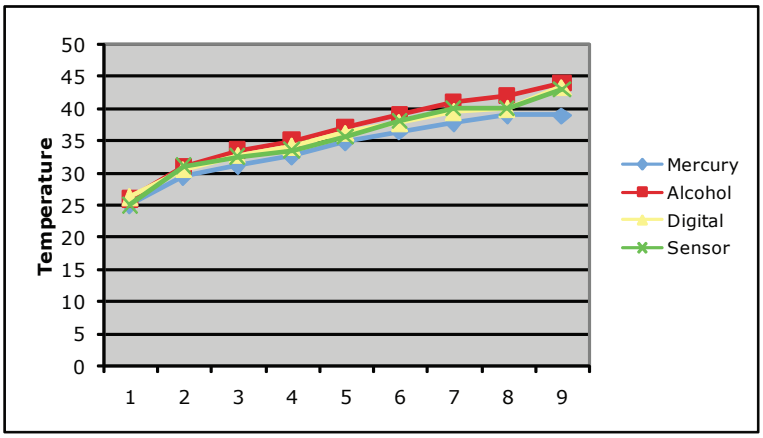

Figure 6.1: The collected measurements of the test.

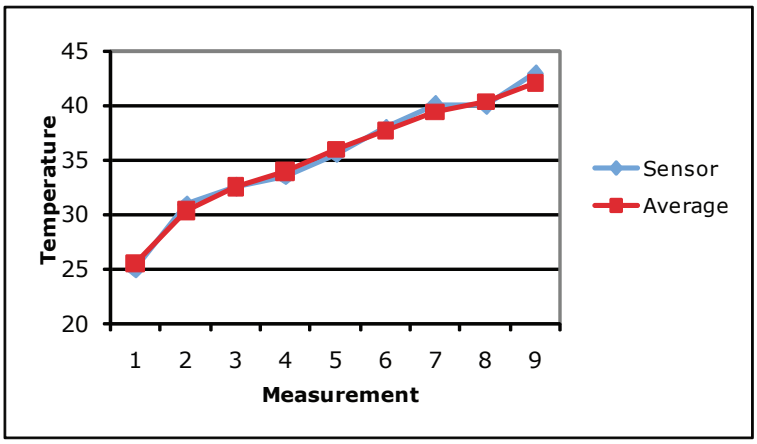

Figure 6.2: The average temperature and the temperature measured by the sensor.

Messurement	Aberration	Unit
1	0.57	°C
2	0.70	°C
3	0.03	°C
4	0.4	°C
5	0.5	°C
6	0.26	°C
7	0.56	°C
8	0.33	°C
9	0.86	°C

Table 6.1: The aberration in relation to the average measurement.

Nr	**Mercury**	**Alcohol**	**Digital**	**Sensor**	Unit
1	25	26	26.3	25	°C
2	29.4	31	30.5	31	°C
3	31.4	33.5	32.7	32.5	°C
4	32.6	35	34.1	33.5	°C
5	34.8	37	36.2	35.5	°C
6	36.4	39	37.8	38	°C
7	37.8	41	39.6	40	°C
8	39	42	40	40	°C
9	39	44	43.4	43	°C

Table 6.2: The actual data the graphs are based upon.

6.1.2 § 3.6.1.2 LCD test

The test seeks to determine, whether all 64 characters of the LCD is working properly.

Test results

The test was executed and all characters work as intended.

6.1.3 § 3.6.1.3 Keypad test

The test seeks to determine, whether all the buttons on the keypad is working properly.

Test results

The test was executed and all buttons work as expected. It is noted, that the # is not working, since the subroutine used to identify pressed buttons does not check for it. This is not a problem, since the # is not used in the software.

6.1.4 § 3.6.1.4 Control unit test

The test seeks to determine, whether the mock system is working properly.

Test results

The test was executed and the mock system works as intended. It can be turned on and off from the software.

6.2 Communication test results

The test journals for the communication tests can be found in appendices C on page 111

6.2.1 § 3.6.2.1 GSM test

The test seeks to determine, whether the communication between the system and the GSM module is working.

Test results

The test was executed and the communication with the GSM module works as intended.
It is noted, that the system did not reply as expected, when the first time the alarm was tested. The problem, that was identified to be a minor error in the system syntax, corrected.

6.2.2 § 3.6.2.2 Local interface test

The test seeks to determine, whether the local interface works properly.

Test results

The test was executed and the interface works as intended. There were glitches, but these were corrected.

6.2.3 § 3.6.2.3 Menu system questionnaire

The results of the questionnaire are missing. This is because the questionnaire was not conducted in the extent, described in the test journal in section B on page 108.
The reason for this is that the preliminary study of the User manual gave rise to many comments and questions. Therefore the Manual was redesigned, to respond to the comments. This however, shifted the time frame of the project and therefore the actual questionnaire could not be preformed.

6.2.4 § 3.6.2.4 Text message syntax questionnaire

The questionnaire was not executed in a adequate way. The reason for this can be seen in 6.2.3.

6.3 System test results

The test journal for the system related tests can be found in appendices D on page 114.

6.3.1 § 3.6.3.1 System update test

The test is preformed to establish, whether the system updates the temperature, time and operating status, at least every 30 seconds.

Test results

The test was executed and the system updates the temperature, time and operating status more than once every 30 seconds. Actually the software updates most of the variables continuously as the main program loop *NORMAL* is executed.

6.3.2 § 3.6.3.2 Administrator alarm test

The test is preformed to demonstrate, that the system can alarm the administrator. Furthermore, the system response time is measured e.g. how long it takes the system to alarm the administrator.

Test results

The test was executed and the results are stated in table 6.3.

Measurements	Alarm time	Units
1	10	s
2	6	s
3	18	s
4	8	s
5	14	s
6	9	s
Average	4.1	s

Table 6.3: The results of the administrator alarm test.

It is noted, the GSM network and not the system software causes most of the delay, in the alarming. As a result, the alarming will vary in response time. The software is designed, so that an alarm message is sent immediately after the measurement of an alarm temperature.

6.3.3 § 3.6.3.3 System endurance test

The test is preformed to demonstrate, that the system can operate for a considerable amount of time without failing.

Test results

The system test is preformed, in accordance with the test journal D. The results are stated in table 6.4.

Time	System status	Reply patteren
11.00	OK	
11.30	System failure	System failure

Table 6.4: The results of the endurance test version I.

The system failed the first test. The failure did not occur, in relation to the sending of commands. The software was examined and the bug identified and corrected. The bug was identified to be an undeleted part of test code, located in the GSM software, which forced the LCD to reset and enter a function loop, from were the system could not return. A new test was preformed, using the new system software. The results are illustrated in table 6.5.

Time	System status	Reply pattern
09.00	OK	
09.30	OK	Error
10.00	OK	OK
10.30	OK	OK
11.00	OK	OK
11.30	OK	OK
12.00	OK	OK
12.30	OK	OK
13.00	OK	OK
13.30	OK	OK
14.00	OK	OK

Table 6.5: The results of the endurance test version II.

It is noted, that the system reply on the first message was not in accordance with the expected. The system was asked to set the set temperature to $35°C$, but the system responded with a "out of range" error message. The problem was after the test, identified to be a wrong default value for the maximum temperature. The maximum was set to $35°C$ which would amount for the error message. The value was changed to $40°C$ in accordance with the performance specifications.
After the test had finished the system was allowed to run freely to see how long a uptime could be maintained. The system suddenly reset at 15.48. The reset was suspicious since all the internal registers of the processor were reset. This normally only happens when the reset button is pressed or the system loses power.

6.4 Summary

From the results of the accept tests, it is concluded that, the system satisfy the demands specified in the performance specifications. This conclusion is drawn, since the product has met all the demands, that have been tested for.
It is noted that the system user manual has not been tested in accordance with the performance specifications. This will be addressed the conclusion in section 7 on the facing page.
Some of the test where conducted several times due to failures in the software. These failures have been corrected and the current software meets the demands of the performance specification. It is however noted that the current software does not support the alarming of the user in the case of the temperature dropping rapidly. This is only a problem if the minimum alarm temperature is close to $0°C$. The problem accrues since the subroutine *ALARM_UPDATE* can not distinguish between positive and negative temperatures. The feature was never implemented since the lowest possible alarm temperature is $0°C$. The problem could be corrected by adding more arguments to the compare function used in the alarm subroutine.
The system record uptime is 9 1/2 hour. The system also reset during this test for some reason unknown to the group.

Chapter 7

Conclusion

In this project, the group has succeeded in designing and constructing a microprocessor based system, capable of monitoring and controlling an electrical device. The system allows both remote and local monitoring and control of the device.
The actual system, is designed around a Motorola 68000 microprocessor. The remote interface consists of a GSM-module and the locale interface consists of a numeric keypad and an LCD. The system monitors the temperature of the surrounding environment, using a temperature sensor. The system simulates the control of an electrical device by turning a LED on and off.

The group has pursued a simple system which illustrates the usability of a microprocessor system in relation to control and monitoring. As a result of this, the system only features a minimum of options and features. This is however not regarded as a problem, since the system is designed to leave room for implementation of further features. The software used in the system is based on a single main program, which calls a series of subroutines that handle different tasks. This setup works well, since it leaves room for implementation of additional features. The implementation of additional features, is however limited by the lack of extra chips selects. The implementation of additional hardware, would call for a redesign of the PEELS and the address space.

The focus of the project has been the implementation and use of a microprocessor system. The group has mapped the specific goals of the semester, to the specific features of the GSM RCS. The main objectives of the project, has been the five specific features described in the system overview 3.2 on page 13.

- Remote control of an electrical device.
- Remote monitoring of an electrical device.
- Autonomous monitoring of an electrical device.
- Autonomous control of an electrical device.
- Remote alarming in relation to the monitoring of an electrical device.

All of the these objectives have been accomplish in this project.
The group is confident in this conviction, since the acceptance test 6 on page 95 confirms, that the demands specified in the performance specifications 3.5 on page 23 have been met.
It is noted, that the acceptance test only verify some of the demands specified in the performance specifications. This is done, since some of the demands are verified during the development of the system. The most basic demands, are thus not mentioned in the acceptance test.

The questionnaire related the system user manual has not been conducted due to a shift in the time frame. This means that it is not possible to verify that the product meet the demands related to the system user manual e.g. the system syntax. A preliminary test has been conducted with a reviewed manual and the feedback has been positive. Still the final test has to been conducted to conclude whether the demands have been met.

The designed software meets the demands specified in the performance specifications, but several steps could be taken to streamline the system and add a few final adjustments. The group has identified a series of software features that could be designed more efficiently. Most of these features however call for a re-design/review of the entire system software and are therefore not pursued further. Some of the additional features are addressed in the future plans 8 on the facing page.

Chapter 8

Future plans

The finished product only features a fraction of the features the GSM RCS, as a commercial product would contain. Several steps could be taken in order to expand the usability of the GSM Remote Control System. Some of these steps are described in the section. The section also addresses the additional features that could be implemented in the system to expand its versatility.

Additional features
As of this moment, the GSM RCS is only designed to monitor one sensor and control one device (the temperature sensor and the mock system).
A way to make the system more versatile, would be to develop the system into a modular system, which allows implementation of several sensors and the control of several devices. This could make the system even more fit for use in holiday homes, since the additional features could be specially designed for this use. Examples of additional implementations could be additional temperature sensors and alarm sensors of different types. The alarm sensors could cover fire alarms and burglar alarms and other safety related sensors.

Sensor mobility
When setting up the GSM RCS system at home, it would be ideal, that temperature sensor could be placed at strategic places in the household. This would call for either a wireless temperature sensor system, or a more versatile connection system. In several new houses a similar system based on PLC technology, is implemented in order to control and monitor the electrical devices in the house. The downside to these systems is, that they are mostly implemented in relation to new construction. In future versions of the GSM RCS, the same approach could be taken, since this would ease the implementation of sensors and the control of several devices in the house. The main focus of this evaluation is the problems related to implementing a wire-based system in an existing house.

EMC
In relation to the EMC problems the GSM RCS does not fully comply with the regulations required to market the device. In fact, the GSM RCS does not or only in some degree comply with the requirements for proper EMC design. Different methods are used to make sure that, a MPU system does not interfere with the surroundings and vice versa. Among others, shielded cables, grounded planes, and twisted cables are used to protect the surrounding environment from the emission of magnetic fields from digital circuits. The shielding of the system from the surrounding environment, which is the most critical problem, when working with digital circuits, is normally build into the casing of the system. This however is only part of the solution since special care to the system environment must be taken into consideration, when design-

ing the actual print of the system.
Long connections must be shielded properly. When working on PCBs this can be done by placing the data lines as far away from the ground, as possible. This can be done in the same manner as on figure 8.1. When shielding a cable that is the source of interference, radiation can be prevented. A conductor carrying

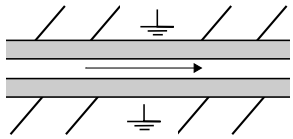

Figure 8.1: Shielding a long conductor.

a current in free air will emit electric and magnetic fields, and cause interference. If the shield is grounded at one point, the electric fields will not come further than the shield. The magnetic fields will, however, still be emitted. By letting a current that is equal and opposite of the conductor, flow in the shield, the magnetic field will be canceled out. This happens because a mutual inductance will occur and neutralize the magnetic fields. Thus will all radiated fields be shielded and not interfere with the surrounding environment [10].
Another way to prevent EMC is to twist the data cables around each other. Datelines, emit magnetic fields, which will influence other nearby circuits and datelines. When twisting two lines, that have data going in each direction, the self-induction will neutralize each other. The same thing can be done on PCBs, by using jumpers. These methods are seen on figure 8.2.

Figure 8.2: Twisted cables.

Gennerel improvements
The GSM RCS is programmed in such a way, that it only allows the use of danish telephone numbers. This is done to insure that the system is not charged extra for sending text messages to foreign phone numbers. At this point, the actual software of the system will not be able to handle foreign phone numbers, but this could pose a problem, since the typical user would be from a different country. This is however a problem the group has omitted, since the designed system is a prototype. In a future version this problem should be taken into consideration.
The same problem is related to the use of a single language, in the menus and the text messages. In future versions the option of selecting a different language could be implemented. Since the text strings are quite common in their structure, it would be easy to have them translated into different languages and add a language menu in the administrator registration.
The system, could also benefit of the implementation of the possibility, of having several administrators, since the current system only support alarming a single administrator e.g. the administrator phone number. There is a number of case, where this could pose a problem, for instants if the administrator does not see the alarm message, or the phone is turned of. This is an acknowledged problem, that in similar systems are solved by adding more administrators to the alarm list.
The current software contains subroutines, which should be reviewed in order to streamline the software. This however calls for a number of serious changes to the system software. In the cases the system is to be further developed these changes should be taken into consideration in order to design a smooth running system.

Bibliography

[1] Alan Clements. *Microprocessor Systems Design*. PWS Publishing Company, third edition, 1997. ISBN 0-534-94822-7.

[2] Fairchild Semiconductor Corporation. 16-key encoder. Datasheet, October 1999.

[3] Seiko Instruments GmbH. Liquid crystal displays - standard character modules - application notes. Datasheet. AN No.SIG-CHMO9805A.

[4] ICT. Peel^{TM} 22cv10a-7/-10/-15/-25, cmos programmable electrically erasable logic device. Datasheet.

[5] Kingbright. T-1 3/4 (5mm) low current led lamps. Datasheet. Komponenten.

[6] Komponenten. Standard keypad. Datasheet.

[7] MAXIM. +5v-powered, multichannel rs-232 driver/receivers. Datasheet, 04 2004. Rev 13.

[8] Motorola. Asyncronous communication interface adapter. Datasheet. http://kom.aau.dk/∼ylm/e4/MC6850.pdf.

[9] Motorola. M68000 8-/16-/32-bit microprocessors user's manual. Datasheet, 1993. Ninth Edition.

[10] Henry W. Ott. *Noise Reduction Techniques in Electronic Systems*. John Wiley & Sons, 2. edition, 1988. ISBN 0-471-85068-3.

[11] Samsung. Samsung k6t4008c1c cmos sram datasheet. Datasheet, April 1999. V 1.0.

[12] National Semiconductor. Precision temperature sensors. Datasheet, May 1999. Komponenten.

[13] National Semiconductor. 3-terminal adjustable current source. Datasheet, March 2000. Komponenten.

[14] National Semiconductor. Adc0820 8-bit high speed μp compatible a/d converter with track/hold function. Datasheet, March 2004. Komponenten.

[15] Siemens. Tc35i at command set. Datasheet, January 2004. Version 02.07.

[16] Siemens. Tc35i hardware interface description. Datasheet, February 2004. Version 02.07.

Appendix A

List of abbreviations and terms

This list contains the abbreviations and terms used in the report. The abbreviations are described only by the full word where terms are described in relation to their meaning in the report.

Abbreviations

- LCD - Liquid Crystal Display
- RAM - Random Access Memory
- ROM - Read Only Memory
- DD RAM - Data display RAM
- CG RAM - Custom character generator RAM
- $\overline{OE}$ - Output enable (active low)
- ADC - Analogue Digital Converter
- LSB - Least Significant Bit
- MPU - Micro Processor Unit
- EMC - ElectroMagnetic Compatibility

Terms

- Alarm Temperature - The term is used to describe two temperatures selected by the user. The temperature consists of a minimum and maximum value that indicates the temperature where the user is to be alerted. If the measured temperature reaches either the minimum or maximum value a alarm code is sent to the user.
- Control Unit - The term is used to describe the device that is used to maintain the selected temperature. The device is not designed by the group.
- SMS syntax - The term is used to describe the standard structure of the text messages in relation to communicating with the system using the GSM-module.
- Least Significant Bit - The bit in the right-hand end of a binary stream.
- Most Significant Bit - The bit in the left-hand end of a binary stream.

- Control state - The term is used to describe the system when it is used as a thermostat.
- Monitoring state - The term is used to describe the system when it only monitors the temperature.
- Set temperature - The term is used to describe the temperature the user or admin has chosen to sustain when the system is in control state.

Appendix B

Testjournal - I/O devices

B.1 Preface

The objective of the test journal is to verify that the I/O devices function as intended. These tests are to be compared to the performance specifications 3.5 on page 23.
The performance specifications are mapped into a test specification, which is used to conduct the actual tests. Based on the results, it is possible to establish, whether the performance specification has been met. The devices and their test paragraph are listed in 3.6.1 on page 26.

B.2 Equipment

The I/O devices use the same equipment for most of the tests, but some extra equipment is needed to test the temperature sensor. The items marked with a * is only used for this test.

Type	**Model**	**AAU nr.**
Power supply	Hameg triple power supply HM7042	33906
Laptop	Webtech	
Adjustable heat source*	<	
Digital Thermometer*		08173
Analog Thermometer (alcohol)*	[-25-115 $^{\circ}C$]	
Analog Thermometer (mercury)*	[-5-101 $^{\circ}C$]	
Ice bucket*		

Table B.1: Equipment for I/O tests

B.3 Setup

The GSM RCS is connected to the power supply and the Webtech laptop is connected to the system, using the serial port for PC communication. The M68k setup file is run and the terminal is setup to communicate on the correct COM port. The power supply is turned on and the TS2 monitor is now communicating with the terminal on the PC. The setup is ready to load programs onto the system and run tests.
For the temperature sensor test, the two analog thermometer, the digital thermometer and the temperature

sensor is placed at the same distance from the heat source.

B.4 Tests

B.4.1 § 3.6.1.1 Temperature measurement I

This test establishes in what degree the temperature sensor can measure the temperature of the surrounding environment. This is done to illustrate, whether the sensor can actually be used to measure the temperature with a sufficient accuracy.

Test procedure

The three thermometers are placed together, with the temperature sensor in the vicinity of an adjustable heat source. The heat source is turned on and the temperature is logged from the three thermometers and the temperature sensor. The temperature is raised and the temperature is logged again. This procedure is repeated 9 times.
It is noted that between the 8th and the 9th measurement, the heat source was moved closer to the setup, since the heat source could not be turned any higher. The distance between the setup and the source was changed from approx 3 cm to 1.5 cm.

B.4.2 § 3.6.1.2 LCD test

The test is designed to test if all 64 characters of the LCD is working properly.

Test procedure

The test is executed by using a test program specificly designed to print character on the LCD. The test program prints the ASCII character #$FF on all characters of the LCD. The #$FF is an ASCII character, that fills all the pixels of the individual characters.
The test program is illustrated below.

```
TEST    NOP
        MOVE.B    #$FF,$900003    PRINT AN FULL BLOCK ON THE LCD
        BRA       TEST            RUN THE FUNCTION CONTUNIUSLY
```

The program code does not feature the initialization of the LCD.

B.4.3 § 3.6.1.3 Keypad test

The test is designed to test if all the buttons on the keypad is working properly.

Test procedure

The test is executed by using a test program specifically designed to print characters on the LCD, when the specific button is pressed. The test program is illustrated below.

```
TEST 2      NOP
            BSR         KEYPAD_C        RUN THE KEYPAD ENTRY SUBROUTINE
            CMP.L       #$”x”, D5
            BEQ         TEST2
            MOVE.B      #$FF,$900003    PRINT THE ENTRY ON THE DISPLAY
            BRA         TEST2
```

The program code does not feature the initialization of the LCD.

B.4.4 § 3.6.1.4 Control unit test

The test is designed to test if the communication with the mock system is working properly.

Test procedure

The test is executed by using two test programs specifically designed to turn the mock system on and off. The two test programs are illustrated below.

The program used to test, if the mock system can be turned on.

```
TEST 3      NOP
            ADDI.B      #0,$D00003      TURN THE MOCKSYSTEM ON
            BRA         TEST2           RUN THE FUNCTION CONTUNIUSLY
```

The program used to test if the mock system can be turned off.

```
TEST 3      NOP
            ADDI.B      #0,$D00001      TURN THE MOCKSYSTEM OFF
            BRA         TEST2           RUN THE FUNCTION CONTUNIUSLY
```

Appendix C

Testjournal - Communication

C.1 Preface

The objective of the test journal, is to test the communication between user/administrator and the GSM RCS. Test paragraphs has been written to test, whether the system meets the demands specified in the performance specification, 3.5 on page 23. The test paragraphs can be found on 3.6.2 on page 27. These paragraphs are not directly executable as tests and therefore, a more specific test description is written to complement each paragraph.

C.2 Equipment

The equipment for the communication tests is basically the same for all the tests, and therefore the equipment is listed as a general equipment list:

Type	**Model**	**AAU nr.**
Power supply	Hameg triple power supply HM7042	33906
Laptop	Webtech	
Mobile phone	Sony Ericsson W880i	

Table C.1: General equipment

C.3 Setup

Because the used equipment is the same for each test, the setup is also similar.
The power supply is connected to the GSM RCS and the laptop is connected to the serial interface on the GSM RCS using a RS-232 cable. The GSM module is connected to the other serial port on the GSM RCS, also using a RS-232 cable. The terminal program on the laptop is setup, using the M68k setup file. The terminal is set to use the correct COM port and afterward the power is turned on and the TS2 monitor is communicating with the terminal on the laptop. The setup is now ready to transfer software to the GSM RCS.

C.4 Tests

C.4.1 § 3.6.2.1 GSM test

This test establish, whether the system can be controlled using the remote interface i.e. the GSM module.

Test procedure

The test it is designed to test four types of commands to the system. The first command tested, is the status command. The second is the set alarm temperature command. The third is the set temperature command and the fourth and final is the outgoing alarm message. The system software, is run from the TS2 monitor and the system program is started up. The person responsible for the test, enters the administrator information, and commences the test. The commands are sent to the system and the system replies are noted to be correct or false. When the alarm message is tested, the administrator enters a alarm temperature low enough to activate the system simply by heating the temperature sensor using the hand.

C.4.2 § 3.6.2.2 Locale interface test

In this test, it is tested whether the the system can be controlled from the local interface e.g. the LCD and keypad.

Test procedure

The test is similar to the GSM test, but only now the test include the use of the local interface. The person responsible for the test commences the test by entering the administrator information. The administrator proceeds to test all the paths described in figure 5.9 on page 85. The test also includes the testing of all of the variables accessible to the administrator and user. These cover the set temperature, the alarm temperature, the operating mode etc.
The test is partly conducted as a part of the debugging of the software but must be executed in one sesion upon completion of the software.

C.4.3 § 3.6.2.3 Menu system questionnaire

This test is preformed to evaluate whether the locale user interface e.g. the menu system, can be rendered intuitive to use. The test is based on a questionnaire.

Test procedure

The test procedure follows the questionnaire design, as seen in section E on page 116.

C.4.4 § 3.6.2.4 Text message syntax questionnaire

This test is preformed,to evaluate whether the text message can be rendered intuitive to use. The test is based on a questionnaire.

Test procedure

The test procedure follows the questionnaire design, as seen in section E on page 116.

Appendix D

Testjournal - System

D.1 Preface

The objective of the test journal is to test the complete system. The tests are compared to the specifications 3.5 on page 23. The completion of the test renders to a conclusion, on whether the performance specification has been met or not.

D.2 Equipment

The I/O devices use the same equipment for most of the tests, but some extra equipment is needed for the test of the temperature sensor. The items marked with a * is only used to test the temperature sensor.

Type	**Model**	**AAU nr.**
Power supply	Hameg triple power supply HM7042	33906
Laptop	Webtech	
Heat source*		
Stop watch*		

Table D.1: Equipment for system tests

D.3 Setup

The GSM RCS is connected to the power supply and the Webtech laptop is connected to the system, using the serial port for PC communication. The M68k setup file is run and the terminal is setup to communicate on the correct COM port. Now the power supply is turned on and the TS2 monitor is communicating with the terminal on the PC. The setup is ready to load programs on to the system and run tests.

D.4 Tests

D.4.1 § 3.6.3.1 System update test

In this test, it is determined whether the system updates the temperature, time and operating status, at least every 30 seconds.

Test procedure

The system software is loaded onto the GSM RCS and set up. The temperature is noted and the heat source is turned on. A stop watch is used to measure, how often the temperature is updated.
Only the temperature is measured, since the update of the temperature is the slowest of the updates. The actual timing is hard to measure since the update happens quite rapidly.

D.4.2 § 3.6.3. Administrator alarm test

In this test, it is tested whether the system can alarm the administrator, if the temperature reaches the alarm temperature.

Test procedure

This test is preformed along the GSM test, in the communication test journal C on page 111

D.4.3 § 3.6.3.3 System endurance test

This test is designed to determine, whether the system can operate for a considerable amount of time without failing.

Test procedure

For this test, the system is set to run in 5 hours. Every hour, the following 4 commands are sent to the system. Set alarm temperature, set temperature (user), status(user) and set alarm(un authorized).
For all the commands sent to the system, the system reply is noted. The reply is dependent on the command, but the actual content (temperature etc.) of the command is not addressed further. The main objective is to test, if the system response as expected, and that the system does not fail.

Appendix E

Questionnaire - Intuitivity test of GSM RCS

E.1 Introduction

This questionnaire is developed to determine the user friendliness of the GSM RCS (GSM Remote Control System). Therefore, some test paragraphs have been developed, which you as participant, are to answer. The questionnaire is estimated to take about 5 minutes of your time. The questions are to be answered on a scale from 1 to 10, where 1 will equal “100 % do not agree” and 10, “100 % agree”.

E.2 Participant information

Please state your gender:
Check the correct box - Female ❑ Male ❑

Please state your age:

_____

E.3 Questions

E.3.1 Local user interface (Menu system)

In this test we would like you to read the user manual. When you have finished reading the user manual, you are ready to answer the following questions:

1. The User manual is easy to understand:
   Please mark the box below the value of your answer.

1	2	3	4	5	6	7	8	9	10

2. The menu system is structured intuitively:

1	2	3	4	5	6	7	8	9	10

E.3.2 SMS syntax

This test is created to determine how intuitive the SMS syntax is for the GSM RCS. Read the SMS syntaxes in the user manual and answer the following question:

1. The SMS syntaxes is intuitive - The SMS commands are easy to understand:

1	2	3	4	5	6	7	8	9	10

E.3.3 Comments

If you have any comments/remarks to the GSM RCS, please feel free to add them on the following lines.

_____
_____
_____
_____
_____

Thank you for your time.

Appendix F

Complete schematic

Appendix G

PEEL software

G.1 PEEL 1

```
module PEEL1;
title 'E4GR415 PEEL 1'
decoder device 'P22V10';

"input pins
A18,A19,A20,A21,A22,A23,AS,LDS,UDS,RW pin 2,3,4,5,6,7,8,9,10,11;
"output pins
CS1,CS2,CS3,CS4,OE,WE pin 14,15,16,17,18,19;

H,L=1,0;

equations

CS1 = A18 # A19 # A20 # A21 # A22 # A23 # UDS;                                    "ROM 1"
CS2 = A18 # A19 # A20 # A21 # A22 # A23 # LDS;                                    "ROM 2"
"CS3 = A20 # A21 # A22 # A23 # !(A18 # A19) # UDS;"
"CS4 = A20 # A21 # A22 # A23 # !(A18 # A19) # LDS;"
CS3 = A23 # A22 # A21 # !((A20 & !(A19 # A18)) # (!A20 & (A19 # A18))) # UDS;    "RAM 1"
CS4 = A23 # A22 # A21 # !((A20 & !(A19 # A18)) # (!A20 & (A19 # A18))) # LDS;    "RAM 2"
OE = !RW # AS;                                                                   "Output Enable"
WE = RW # AS;                                                                    "Write Enable"

test_vectors
([A23,A22,A21,A20,A19,A18,AS,LDS,UDS,RW] -> [CS1,CS2,CS3,CS4,OE,WE])
 "CHIPSELECT 1"
 [ L,L,L,L,L,L,.X.,H,L,.X.] -> [ L,H,H,H,.X.,.X.];
 "CHIPSELECT 2"
 [ L,L,L,L,L,L,.X.,L,H,.X.] -> [ H,L,H,H,.X.,.X.];
 "CHIPSELECT 3"
 [ L,L,L,L,L,H,.X.,H,L,.X.] -> [ H,H,L,H,.X.,.X.];
 [ L,L,L,L,H,L,.X.,H,L,.X.] -> [ H,H,L,H,.X.,.X.];
 [ L,L,L,L,H,H,.X.,H,L,.X.] -> [ H,H,L,H,.X.,.X.];
 [ L,L,L,H,L,L,.X.,H,L,.X.] -> [ H,H,L,H,.X.,.X.]; "Wraparound test"
 [ L,L,L,H,H,H,.X.,H,L,.X.] -> [ H,H,H,H,.X.,.X.]; "Error test, CS3 cannot be active here!"
 "CHIPSELECT 4"
 [ L,L,L,L,L,H,.X.,L,H,.X.] -> [ H,H,H,L,.X.,.X.];
 [ L,L,L,L,H,L,.X.,L,H,.X.] -> [ H,H,H,L,.X.,.X.];
 [ L,L,L,L,H,H,.X.,L,H,.X.] -> [ H,H,H,L,.X.,.X.];
 [ L,L,L,H,L,L,.X.,L,H,.X.] -> [ H,H,H,L,.X.,.X.]; "Wraparound test"
 [ L,L,L,H,H,H,.X.,L,H,.X.] -> [ H,H,H,H,.X.,.X.]; "Error test, CS4 cannot be active here!"
```

```
 "OUTPUT␣ENABLE␣&␣WRITE␣ENABLE"
 [ .X.,.X.,.X.,.X.,.X.,L,.X.,.X.,H] -> [ .X.,.X.,.X.,.X.,L,H];
 [ .X.,.X.,.X.,.X.,.X.,L,.X.,.X.,L] -> [ .X.,.X.,.X.,.X.,H,L];
 "Error␣test
␣[␣H,L,H,H,L,L,.X.,L,H,.X.]␣->␣[␣H,H,H,H,.X.,.X.];␣"Error test
 [ H,L,H,H,L,L,.X.,H,L,.X.] -> [ H,H,H,H,.X.,.X.]; "Error␣test
end␣PEEL1;
```

G.2 PEEL 2

```
module PEEL2;
title 'E4GR415␣PEEL␣2'
decoder device 'P22V10';

"input␣pins
A18,A19,A20,A21,A22,A23,AS,LDS,VMA,CSI7,CSI8,INT␣pin␣2,3,4,5,6,7,8,9,10,11,13,20;
"output pins
CS5,CS6,CS7,CS8,CS9,CS10,DTACK,VPA,nCS7 pin 14,15,16,17,18,19,22,23,21;

H,L=1,0;
X=.X.;

equations

CS5 = !A23 # A20 # A21 # A22 # VMA;                "UART"
CS6 = A23 & A20 & !(A21 # A22) & !LDS;             "LCD"
CS7 = !(A23 & A21) # A20 # A22 # LDS;       "Keypad"
nCS7 = !CS7;
CS8 = !(A23 & A21 & A20) # A22 # VMA;              "GSM"
CS9 = !(A23 & A22) # A20 # A21 # LDS;              "ADC"
CS10 = !(!(A23 & A22 & A20) # A21 # LDS);             "Control␣Unit"
DTACK = AS # (!CS9 & INT);
"VPA␣=␣!A23␣#␣A20␣#␣A21␣#␣A22␣#␣AS␣#␣LDS;
VPA␣=␣(!A23␣#␣A20␣#␣A21␣#␣A22␣#␣AS␣#␣LDS)␣&␣(!(A23␣&␣A21␣&␣A20)␣#␣A22␣#␣AS␣#␣LDS)␣&␣CSI8␣&␣CSI7;

test_vectors
([A23,A22,A21,A20,A19,A18,LDS,VMA,AS,INT]␣->␣[CS5,CS6,CS7,CS8,CS9,CS10,VPA,DTACK])

␣[H,L,L,L,L,L,L,L,L,X]␣->␣[L,L,H,H,H,L,X,L];␣"TS2
 [H,L,L,H,L,L,L,X,L,X] -> [H,H,H,H,H,L,X,L]; "LCD
␣[H,L,H,L,L,L,L,X,L,X]␣->␣[H,L,L,H,H,L,X,L];␣"KEYB
 [H,L,H,H,L,L,L,L,L,X] -> [H,L,H,L,H,L,X,L]; "GSM
␣[H,H,L,L,L,L,L,X,L,X]␣->␣[H,L,H,H,L,L,X,L];␣"ADC
 [H,H,L,H,L,L,L,X,L,X] -> [H,L,H,H,H,H,X,L]; "Control

␣[H,L,L,L,L,L,L,L,L,X]␣->␣[L,L,H,H,H,L,L,X];␣"VPA KEYB
 [H,L,H,H,L,L,L,L,L,X] -> [H,L,H,L,H,L,L,X]; "VPA␣GSM
␣[H,H,L,L,L,L,L,X,L,H]␣->␣[H,L,H,H,L,L,X,H];␣"DTACK ADC HØJ (ADC ikke klar)
end PEEL2;
```

G.3 PEEL 3

```
module PEEL3;
title 'E4GR415␣PEEL␣3'
decoder device 'P22V10';

"input␣pins
A1,A2,A3,NMI1,NMI2,IRL5,IRL4,FC0,FC1,FC2␣pin␣2,3,4,5,6,7,8,9,10,11;
"output pins
IPL0,IPL1,IPL2,CSI8,CSI7,NMI pin 14,15,16,17,18,19;
```

```
H,L=1,0;
X=.X.;

IPL= [IPL2,IPL1,IPL0];
FC= [FC2,FC1,FC0];

ADDR = [A3,A2,A1];

equations

NMI = NMI1 & NMI # !NMI2;

[IPL2,IPL1,IPL0] = !([L,L,L] # ([H,H,H] & NMI) # ([H,L,H] & !IRL5) # ([H,L,L] & !IRL4 & IRL5));
CSI8 = !((ADDR==[H,L,L]) & (FC==[H,H,H]));
CSI7 = !((ADDR==[H,L,H]) & (FC==[H,H,H]));

test_vectors
([A3,A2,A1,NMI1,NMI2,IRL4,IRL5,FC2,FC1,FC0] -> [IPL2,IPL1,IPL0,CSI8,CSI7])
 "NMI"
 [X,X,X,H,L,H,H,X,X,X] -> [L,L,L,H,H];
 "IRL␣5,␣IRQ␣Keyp"
 [X,X,X,L,H,H,L,L,L,L] -> [L,H,L,H,H];
 "IRL␣4,␣IRQ␣GSM"
 [X,X,X,L,H,L,H,L,L,L] -> [L,H,H,H,H];
 "CSI8,␣IRA␣GSM"
 [H,L,L,L,H,L,H,H,H,H] -> [L,H,H,L,H];
 "CSI7,␣IRA␣Keyb"
 [H,L,H,L,H,H,L,H,H,H] -> [L,H,L,H,L];
 "Prioriteret␣interupt␣keyb␣med␣2.␣prioritet,␣NMI␣1.␣prioritet
␣[H,L,H,L,H,L,L,H,H,H]␣->␣[L,H,L,H,L];
␣[X,X,X,H,L,L,L,X,X,X]␣->␣[L,L,L,H,H];
end␣PEEL3;
```

Appendix H

GSM software

H.1 Main Functions

```
*********************************************
INIT_GSM        NOP                             INITALIZING THE GSM MODULE
                ANDI.L          #$00FFFFFF,D4   CLEARS THE USED STATUS BITS
                MOVE.B          #3,GSM_SR       RESET THE ACIA
                MOVE.B          ACIA_I,GSM_SR   SET UP THE ACIA
                BSR             MOVE_BUF        RESET THE BUFFER
****
INIT_GSM_01     LEA             E0,A0           LOAD COMAND ADDRESS
                BSR             SEND_ST_GSM     SUBRUTINE CALL TO SEND COMMAND
                BSR             WAIT2LF         WAITING FOR THE RESPONS TO BE IN THE BUFFER
                BSR             CMP_ANS         SUBRUTINE TO RECONISE RESPONSE
                BSR             MOVE_BUF        RESET BUFFER
                BTST.L          #31,D4          CONTROL IF THE ANSWER WAS CORECT
                BEQ             INIT_GSM_01     IF NOT, JUMPS
                BCLR.L          #31,D4          RESET STATUS BIT

****
INIT_GSM_02     LEA             CPIN,A0         LOAD COMAND ADDRESS
                BSR             SEND_ST_GSM     SUBRUTINE CALL TO SEND COMMAND
                BSR             WAIT2LF         WAITING FOR THE RESPONS TO BE IN THE BUFFER
                BSR             CMP_ANS         SUBRUTINE TO RECONISE RESPONSE
                BSR             MOVE_BUF        RESET BUFFER

                BTST.L          #30,D4          CONTROLE IF WRONG PIN
                BEQ             INIT_GSM_06     JUMP IF RIGHT PIN
                BSR             P_ADMINREG4     CALL WRONG PIN MENU TO LCD
INIT_GSM_06     BCLR.L          #30,D4          RESET STATUS BIT

                BTST.L          #29,D4          CONTROLES IF SIM NEED PUK
                BEQ             INIT_GSM_07     JUMP IF NOT
*                BSR             ENTER_PUK       FUNCTION TO ENTER PUK, NOT INCLUDED!
INIT_GSM_07     BCLR.L          #29,D4          RESET STATUS BIT

                BTST.L          #31,D4          CONTROL IF THE ANSWER WAS CORECT
                BEQ             INIT_GSM_02     IF NOT, JUMPS
                BCLR            #31,D4          RESET STATUS BIT
*****
INIT_GSM_03     LEA             CMGF,A0         LOAD COMAND ADDRESS
                BSR             SEND_ST_GSM     SUBRUTINE CALL TO SEND COMMAND
                BSR             WAIT2LF         WAITING FOR THE RESPONS TO BE IN THE BUFFER
```

```
                BSR             CMP_ANS         SUBRUTINE TO RECONISE RESPONSE
                BSR             MOVE_BUF        RESET BUFFER

                BTST.L          #31,D4          CONTROL IF THE ANSWER WAS CORECT
                BEQ             INIT_GSM_03     IF NOT, JUMPS
                BCLR            #31,D4          RESET STATUS BIT
****
INIT_GSM_04     LEA             CNMI,A0         LOAD COMAND ADDRESS
                BSR             SEND_ST_GSM     SUBRUTINE CALL TO SEND COMMAND
                BSR             WAIT2LF         WAITING FOR THE RESPONS TO BE IN THE BUFFER
                BSR             CMP_ANS         SUBRUTINE TO RECONISE RESPONSE
                BSR             MOVE_BUF        RESET BUFFER

                BTST.L          #31,D4          CONTROL IF THE ANSWER WAS CORECT
                BEQ             INIT_GSM_04     IF NOT, JUMPS
                BCLR            #31,D4          RESET STATUS BIT
****
INIT_GSM_05     LEA             CCLK,A0         LOAD COMAND ADDRESS
                BSR             SEND_ST_GSM     SUBRUTINE CALL TO SEND COMMAND
                BSR             WAIT2LF         WAITING FOR THE RESPONS TO BE IN THE BUFFER
                BSR             CMP_ANS         SUBRUTINE TO RECONISE RESPONSE
                BSR             MOVE_BUF        RESET BUFFER

                BTST.L          #29,D4          CONTROL IF TIME VALID
                BEQ             INIT_GSM_08     JUMP IF
                BSR             P_ADMINREG5     CALL WRONG TIME MENU TO LCD
                BCLR.L          #29,D4
INIT_GSM_08     BTST.L          #31,D4          CONTROL IF THE ANSWER WAS CORECT
                BEQ             INIT_GSM_05     IF NOT, JUMPS
                BCLR            #31,D4          RESET STATUS BIT

                RTS                             RETURN TO PREVIOUS FUNCTION
***********************************************
CMP_ANS         NOP                             RECONISING ANSWERS
                LEA             ELM0,A1         SET POINTER TO 1. ELEMENT
CMP_ANS0        MOVE.L          (A1),A2         SET POINTER TO DATA STRING
                MOVE.L          A4,A3           MAKING COPY OF BUFFER START POINTER

CMP_ANS1        CMP.B           #0,(A2)         CMP IF STRING ARE ET THE END
                BEQ             END_V           IF VALID, JUMPS
                CMP.L           A5,A3           CMP IF RETCHED END OF BUFFER
                BEQ             END_NK          JUMPS IF VALID

                MOVE.B          (A3)+,D0        MOVE RECIVED VALU TO REGISTER
                CMP.B           (A2)+,D0        CMP REGISTER TO DATA STRING
                BNE             CMP_ANS2        JUMP IF
                BRA             CMP_ANS1        JUMP BACK AND START OVER WITH NEW CHAR

CMP_ANS2        CMP.L           #0,8(A1)        LOOK IF LAST ELEMENT
                BEQ             END_NK          JUMP IF
                MOVE.L          8(A1),A1        LOAD NEXT ELEMENT
                BRA             CMP_ANS0        START OVER
*****
END_NK          NOP                             NOT KNOWN COMMAND
                BSET.L          #28,D4          SET TEST BIT
                BSR             MOVE_BUF        RESET BUFFER
                RTS
*****
END_V           NOP                             VALID COMMAND
                MOVE.L          4(A1),A6        COPY SUBROUTINE ADDRES
                JSR             (A6)            EXECUTE SUBROUTINE
                BSR             MOVE_BUF        RESET BUFFER
                RTS                             RETURN TO PREVIOUS FUNCTION
***********************************************
```

```
GSM_UP          NOP                             UPDATE GSM NETWORK STATUS
                ADD.B           #1,GSMCOUNTER   COUNTING ATEMPT
                CMP.B           #$FF,GSMCOUNTER CMP IF IT TIME TO UPDATE
                BNE             GSM_UP_01       JUMP IF NOT

GSM_UP_02       BSR             MOVE_BUF        RESET BUFFER
                BCLR.L          #28,D4          CLEAR STATUS BIT
                LEA             CSQ,A0          LOAD COMMAND POINTER
                BSR             SEND_ST_GSM     SEND COMMAND
                BSR             WAIT4LF         WAIT FOR 4 LINE FEEDS
                BSR             CMP_ANS         RECONISE ANSWER
                BSR             MOVE_BUF        RESET BUFFER
                BTST.L          #28,D4          CONTROLE STATUS BIT
                BNE             GSM_UP_02       IF NOT JUMP
                MOVE.B          #0,GSMCOUNTER   RESET ATEMPT COUNTER
GSM_UP_01       RTS                             RETURN
***********************************************
CLOCK_UP        NOP                             UPDATE TIM
                ADD.B           #1,CLKCOUNTER   ADD ATEMPT
                CMP.B           #$FF,CLKCOUNTER LOOK IF ITS TIME
                BNE             CLOCK_UP_01     IF NOT JUMP
CLOCK_UP_02     BSR             MOVE_BUF        RESET BUFFER
                BCLR.L          #28,D4          RESET STATUS BIT
                LEA             CCLK_Q,A0       LOAD COMMAND POINTER
                BSR             SEND_ST_GSM     SEND COMMAND
                BSR             WAIT4LF         WAIT FOR 2 LINE FEEDS
                BSR             CMP_ANS         RECONISE RESPONSE
                BSR             MOVE_BUF        RESET BUFFER
                MOVE.B          #0,CLKCOUNTER   RESET ATEMPT COUNTER
                BTST.L          #28,D4          CONTROLATE STATUS BIT
                BNE             CLOCK_UP_02     F NOT, JUMP
CLOCK_UP_01     RTS                             RETURN
***********************************************
MSG_UP          CMP.B           #0,INCOUNTER    CMP TO LOOK FOR RESPONSE
                BEQ             MSG_UP0         IF NOT JUMP
                BSR             WAIT2LF         WAIT FOR 2 LINE FEEDS
                BSR             CMP_ANS         RECONISE ANSWER
                BSR             MOVE_BUF        RESET BUFFER
MSG_UP0         RTS                             RETURN
***********************************************
INPUT_GSM       NOP                             INTERRUPT FUNCTION FOR GSM
                MOVE.L          D0,-(A7)        SAVE REGISTER ON STACK
                MOVE.B          GSM_DR,D0       RECIVE CHAR
                CMP.B           #$0A,D0         CMP TO FINDE LINE FEED
                BNE             INPUT_GSM1      JUMP IF NOT
                MOVE.L          #0,D1           RESET REGISTER
                MOVE.B          INCOUNTER,D1    MOVE LF COUNTER TO REGISTER
                ADD.B           #1,D1           COUNT 1 LINE FEED
                MOVE.B          D1,INCOUNTER    SAVE REGISTER IN VARIABEL
INPUT_GSM1      MOVE.B          D0,(A5)+        MOVES CHAR TO BUFFER
                MOVE.L          (A7)+,D0        RESSTORE REGISTER
                RTE                             RETURN
***********************************************
INIT_BUFFER     NOP                             INITILAIZE BUFFER
                LEA         BUF_S_POINT,A4      LOADS START OF BUFFER
                LEA         (BUF_S_POINT+300),A5 LOADS MAX SIZE OF BUFFER
INIT_BUFFER0    CMP.L       A5,A4               CMP START AND END
                BEQ         INIT_BUFFER1        JUMP IF SIMULAR
                MOVE.B      #0,(A4)+            RESET MEMORY
                BRA         INIT_BUFFER0        JUMP TO NEXT ADDRESS
INIT_BUFFER1    RTS                             RETURN
***********************************************
```

H.2 Subroutines

```
*********************************************
WAIT1LF     NOP                             WAIT 1 LINE FEED
            CMP.B       #1,INCOUNTER        CMP LF COUNTER
            BNE         WAIT1LF             JUMP IF NOT
            MOVE.B      #0,INCOUNTER        RESET LF COUNTER
            BSR         WAIT                WAIT!
            RTS                             RETURN
*********************************************
WAIT2LF     NOP                             WAIT 1 LINE FEED
            CMP.B       #2,INCOUNTER        CMP LF COUNTER
            BNE         WAIT2LF             JUMP IF NOT
            MOVE.B      #0,INCOUNTER        RESET LF COUNTER
            BSR         WAIT                WAIT!
            RTS                             RETURN
*********************************************
WAIT4LF     NOP                             WAIT 1 LINE FEED
            CMP.B       #4,INCOUNTER        CMP LF COUNTER
            BNE         WAIT4LF             JUMP IF NOT
            MOVE.B      #0,INCOUNTER        RESET LF COUNTER
            BSR         WAIT                WAIT!
            RTS                             RETURN
*********************************************
WAIT5LF     NOP                             WAIT 1 LINE FEED
            CMP.B       #5,INCOUNTER        CMP LF COUNTER
            BNE         WAIT5LF             JUMP IF NOT
            MOVE.B      #0,INCOUNTER        RESET LF COUNTER
            BSR         WAIT                WAIT!
            RTS                             RETURN
*********************************************
SEND_ST_GSM NOP                             THE SUBRUTINE TO SEND SIMPLE COMMANDS TO GSM
            BTST.B      #1,GSM_SR           TEST THE TRANSMITTER STATUS
            BEQ         SEND_ST_GSM         POLL UNTIL TRANSMITTER READY FOR DATA
            CMP.B       #00,(A0)            CONTROLE IF THE COMMAND ARE AT THE END
            BEQ         SEND_ST_GSM1        JUMPS TO NEXT CHAR,
            MOVE.B      (A0)+,GSM_DR        MOVES ONE CHAR TO THE GSM MODULE AT A TIME
            BRA         SEND_ST_GSM         REAPEATS WHIT THE NEXT CHAR
SEND_ST_GSM1 RTS                            END OF SEND COMMAND
*********************************************
READ_MSG    NOP                             FIND SMS (CMP_ANS TRIGGERD)
            ANDI.L      #$FF0FFFFF,D4       RESET THE USER STATUS BITS
            MOVE.L      A4,A3               MAKE COPY OF BUFER START POINTER
            MOVE.L      #0,D0               RESET REGISTER

ACTION6_04  ADD.L       #1,D0               COUNT
            ADD.L       #1,A3               MOVE POINTER
            CMP         #14,D0              CMP AMOUNT OF MOVES
            BNE         ACTION6_04          IF NOT, MOVE AGAIN
            MOVE.B      (A3),MSG_NR         SAVE SMS STORAGE NUMBER
            BSR         READ_SMS            SUBRUTINE TO READ SMS

            BTST.L      #21,D4              CONTROLE SENDER STATUS
            BNE         USE_NOUSER          JUMP IF RIGHT BIT
            BRA         USE_US_AD           JUMP TO ACCEPTED USER

ACTION603   BSR         DELET_SMS           SUBRUTINE TO DELET SMS
            RTS                             RETURN

USE_NOUSER  BSR         MOVE_BUF            RESET BUFFER
            MOVE.B      #6,SEND_MSG         SELECT UNARTUSER SMS
            BSR         SEND_SMS            SUBRUTINE TO SEND SMS
            BRA         ACTION603           JUMP
```

```
USE_US_AD   BSR         FIND_MSG        SUBRUTINE TO FINDE SMS
            BSR         MOVE_BUF        RESET BUFFER
            BRA         ACTION603       JUMP
*********************************************
SEND_SMS    NOP                         SENDS SMS
            BSR         MOVE_BUF        RESET BUFFER
            LEA         SEND,A0         SET COMMAND POINTER
            BSR         SEND_ST_GSM     SEND COMMAND
            LEA         TEMP_NR,A0      *
            BSR         SEND_ST_GSM     *
            LEA         COM_END,A0      *
            BSR         SEND_ST_GSM     *
            BSR         WAIT1LF         WAIT FOR 1 LINE FEED
            BSR         CMP_ANS         RECONISE ANSER
            BSR         WAIT2LF         WAIT 2 LINE FEED
            BSR         CMP_ANS         RECONISE RESPOND
            BSR         MOVE_BUF        RESET BUFFER
            RTS                         RETURN
*********************************************
READ_SMS    NOP                         READ SMS
            BSR         MOVE_BUF        RESET BUFFER
            LEA         READ,A0         SET COMMAND POINTER
            BSR         SEND_ST_GSM     SEND THE COMMAND
            LEA         MSG_NR,A0       *
            BSR         SEND_ST_GSM     *
            LEA         COM_END,A0      *
            BSR         SEND_ST_GSM     *
            BSR         WAIT5LF         WAIT OR 5 LINE FEEDS
            BSR         FIND_NR         SUBRUTINE TO FINDE PHONE NUMBER
            BSR         CMP_NR          SUBRUTINE TO CONTROLATE SENDER
            RTS                         RETURN
*********************************************
DELET_SMS   NOP                         DELET SMS
            BSR         MOVE_BUF        RESET BUFFER
            LEA         DELET,A0        LOAD COMMAND POINTER
            BSR         SEND_ST_GSM     SEND COMMAND
            LEA         MSG_NR,A0       *
            BSR         SEND_ST_GSM     *
            LEA         COM_END,A0      *
            BSR         SEND_ST_GSM     *
            BSR         WAIT2LF         WAIT FOR 2 LINE FEEDS
            BSR         CMP_ANS         RECONISING RESPOND
            BSR         MOVE_BUF        RESET BUFFER
            RTS                         RETURN
*********************************************
FIND_NR     NOP                         FIND PHONE NUMBER
            MOVE.L      #0,D0           RESET REGISTER
            MOVE.L      #0,D1           RESET REGISTER
            LEA         TEMP_NR,A0      LOAD POINTER TO SAVE NUMBER
            MOVE.L      A4,A3           COPY START POINTER
LOOK_NR     CMP.L       A3,A5           LOOK OF AT END OF BUFFER
            BEQ         END_NR          JUMP IF
            MOVE.B      (A3)+,D0        MOVE CHAR TO BUFFER
            CMP.B       #',',D0         COUNT ,
            BEQ         ADD_K           JUMP IF
            CMP.B       #1,D1           CMP AMOUNT OF , BEEN
            BNE         LOOK_NR         IF NOT RIGHT JUMP TO NEXT CHAR
            MOVE.B      D0,(A0)+        MOVE CHAR TO VARIABEL
            BRA         LOOK_NR         JUMP

END_NR      RTS                         RETURN

ADD_K       ADD.B       #1,D1           COUNT ,
```

```
                BRA             LOOK_NR             JUMP
*********************************************
CMP_NR          NOP                                 FIND USER OR ADMIN
                LEA             TEMP_NR,A0          SET POINTER TO TEMPERY NUMBER
                LEA             ADMIN_NR2,A1        SET POINTER TO ADMIN NR

CN1             CMP.B           #0,(A0)             COM IF END OF NR
                BEQ             GO_ADMIN            IF TRUE JUMP
                MOVE.B          (A0)+,D0            MOVE CHAR TO REGISER
                CMP.B           (A1)+,D0            CMP TEMP AND ADMIN
                BNE             LOOK_USER           IF NOT SIMULAR JUMP
                BRA             CN1                 JUMP

LOOK_USER       LEA             TEMP_NR,A0          SET POINTER TO TEMP NR
                LEA             USER_NR2,A1         SET POINTER TO USER NR
CN2             CMP.B           #0,(A0)             CMP IF END OF NR
                BEQ             GO_USER             JUMP IF TRUE
                MOVE.B          (A0)+,D0            MOVE CHAR TO REGISTER
                CMP.B           (A1)+,D0            CMP TEMP AND USER
                BNE             GO_NOUSER           IF NOT SIMULAR JUMP
                BRA             CN2                 JUMP

GO_ADMIN        BSET.L          #23,D4              SET BIT FOR ADMIN
                RTS                                 RETURN
GO_USER         BSET.L          #22,D4              SET BIT FOR USER
                RTS                                 RETURN
GO_NOUSER       BSET.L          #21,D4              SET BIT FOR UNKNOWN SENDER
                RTS                                 RETURN
*********************************************
FIND_MSG        MOVE.L          A4,-(A7)            SAVES THE BUFFER START POINTER ON STACK
                MOVE.L          #0,D0               RESET REGISTER
                MOVE.L          #0,D1               RESET REGISTER
                MOVE.L           A4,A3              COPY BUFFER POINTER
FIND_MSG1       CMP.L            A3,A5              CMP IF END OF BUFFER
                BEQ              FIND_MSG3          JUMP IF
                MOVE.B          (A3)+,D0            MOVE CHAR TO REGISTER
                CMP.B            #$0A,D0            LOOK FOR LINE FEED
                BNE              FIND_MSG1          JUMP IF NOT
                CMP.B            #1,D1              CMP AMOUNT OF LF
                BEQ              FIND_MSG4          IF RIGHT JUMP
                ADD.L            #1,D1              COUNT LF
                BRA              FIND_MSG1          JUMP

FIND_MSG3       BSR              CMP_ANS            RUN SUBROUTINE TO RECONISE SMS
                BTST.L           #28,D4             CONTROLES IF SMS KNOWN
                BNE              FIND_MSG5          JUMP IF NOT
                MOVE.L          (A7)+,A4            RESTORE BUFFER POINTER
                RTS                                 RETURN

FIND_MSG4       MOVE.L           A3,A4              MOVE BUFFER POINTER TO SMS START
                BRA              FIND_MSG3          JUMP

FIND_MSG5       MOVE.B           #8,SEND_MSG        SELECT SYNTAX ERROR SMS
                BSR              SEND_SMS           SEND SMS
                MOVE.L          (A7)+,A4            RESTORE BUFFER POINTER
                RTS                                 RETURN
*********************************************
ACTION0         NOP                                 SET OK BIT (CMP_ANS TRIGGERD)
                BSET.L           #31,D4             SET BIT IN REGISTER
                RTS                                 RETURN
*********************************************
ACTION3         NOP                                 SET WRONG PIN BIT (CMP_ANS TRIGGERD)
                BSET.L           #30,D4             SET BIT IN REGISTER
                RTS                                 RETURN
```

```
*********************************************
ACTION4         NOP                             SET WRONG INDEX BIT (CMP_ANS TRIGGERD)
                BSET.L          #29,D4          SET BIT IN REGISTER
                RTS                             RETURN
*********************************************
ACTION5         NOP                             SET SIM BUSY BIT (CMP_ANS TRIGGERD)
                BSET.L          #26,D4          SET BIT IN REGISTER
                RTS                             RETURN
*********************************************
ACTION7         NOP                             SMS SENT (CMP_ANS TRIGGERD)
*           MESSAGES SENT                        NOT USED IN THE VERSION
                RTS                             RETURN
*********************************************
ACTION8         NOP                             GSM NETWORK STATUS (CMP_ANS TRIGGERD)
                MOVE.L          #0,D0           RESET REGISTER
                MOVE.L          #0,D1           RESET REGISTER

ACTION8_01      CMP.B           #$30,(A3)       CMP TO FIND ASCII NUMBERS
                BLT             ACTION8_07      JUM IF NOT
                CMP.B           #$39,(A3)       *
                BGT             ACTION8_07      *

                ADD.B           #1,D0           COUNT AMOUNT OF NUMBERS FOUND
                ROL.L           #8,D1           ROTATE REGISTER
                MOVE.B          (A3),D1         SAVE NUMBER IN REGISTER
ACTION8_07      ADD.L           #1,A3           MOVING POINTER
                CMP.B           #4,D0           LOOK IF ALL NUMBERS FOUND
                BNE             ACTION8_01      JUMP IF NOT

                SWAP.W          D1              SWAP WORDS IN REGISTER
                CMP.W           #$3939,D1       LOOK FOR SIGNAL STRENGT
                BEQ             ACTION8_02      IF NONE JUMP
                BSET.L          #1,D4           SET STATUS BIT
                RTS                             RETURN

ACTION8_02      BCLR.L          #1,D4           CLEAR STATUS BIT
                RTS                             RETURN
*********************************************
ACTION9         NOP                             TIME UPDATE (CMP_ANS TRIGGERD)
                MOVE.L          #0,D0           RESET REGISTER
ACTION9_01      CMP.B           #$30,(A3)       CMP TO FIND ASCII NUMBERS
                BLT             ACTION9_07      JUMP IF NOT
                CMP.B           #$39,(A3)       *
                BGT             ACTION9_07      *
                ADD.B           #1,D0           COUNT NUMBERS
                CMP.B           #7,D0           CMP AMOUNT OF NUMBERS
                BLT             ACTION9_07      IF LESS JUMP
                CMP.B           #10,D0          CMP AMOUNT OF NUMBER
                BGT             ACTION9_07      IF MORE JUMP
                ROL.L           #8,D6           ROTATE TIME REGISTER TO FIT NUMBER
                MOVE.B          (A3),D6         MOVE NUMBER TO TIME REGISTER
ACTION9_07      ADD.L           #1,A3           MOVE POINTER
                CMP.B           #10,D0          CMP AMOUNT OF NUMBERS TO SE IF END
                BNE             ACTION9_01      JUM IF NOT
                RTS                             RETURN
*********************************************
ACTION10        NOP                             READY SIGNAL TO TRANSMIT SMS (CMP_ANS TRIGGERD)
                CMP.B           #0,SEND_MSG     CMP IF SMS CHOSEN
                BEQ             ACTION10_13     JUMP IF NOT

                CMP.B           #1,SEND_MSG     CMP NR OF CHOSEN SMS
                BNE             ACTION10_02     JUMP TO NEXT IF NOT THIS
                BSR             ALARENAB        RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG     RESET SMS CHOISE
```

```
                BRA             ACTION10

ACTION10_02     CMP.B           #2,SEND_MSG     CMP NR OF CHOSEN SMS
                BNE             ACTION10_03     JUMP TO NEXT IF NOT THIS
                BSR             ALARDISA        RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG     RESET SMS CHOISE
                BRA             ACTION10        JUMP TO END

ACTION10_03     CMP.B           #3,SEND_MSG     CMP NR OF CHOSEN SMS
                BNE             ACTION10_04     JUMP TO NEXT IF NOT THIS
                BSR             CTRLENAB        RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG     RESET SMS CHOISE
                BRA             ACTION10        JUMP TO END

ACTION10_04     CMP.B           #4,SEND_MSG     CMP NR OF CHOSEN SMS
                BNE             ACTION10_05     JUMP TO NEXT IF NOT THIS
                BSR             CTRLDISA        RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG     RESET SMS CHOISE
                BRA             ACTION10        JUMP TO END

ACTION10_05     CMP.B           #5,SEND_MSG     CMP NR OF CHOSEN SMS
                BNE             ACTION10_06     JUMP TO NEXT IF NOT THIS
                BSR             STATUS          RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG     RESET SMS CHOISE
                BRA             ACTION10        JUMP TO END

ACTION10_06     CMP.B           #6,SEND_MSG     CMP NR OF CHOSEN SMS
                BNE             ACTION10_07     JUMP TO NEXT IF NOT THIS
                BSR             ERUNAUTH        RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG     RESET SMS CHOISE
                BRA             ACTION10        JUMP TO END

ACTION10_07     CMP.B           #7,SEND_MSG     CMP NR OF CHOSEN SMS
                BNE             ACTION10_08     JUMP TO NEXT IF NOT THIS
                BSR             TEMPCONF        RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG     RESET SMS CHOISE
                BRA             ACTION10        JUMP TO END

ACTION10_08     CMP.B           #8,SEND_MSG     CMP NR OF CHOSEN SMS
                BNE             ACTION10_09     JUMP TO NEXT IF NOT THIS
                BSR             ERSYNTAX        RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG     RESET SMS CHOISE
                BRA             ACTION10        JUMP TO END

ACTION10_09     CMP.B           #9,SEND_MSG     CMP NR OF CHOSEN SMS
                BNE             ACTION10_10     JUMP TO NEXT IF NOT THIS
                BSR             ALARCONF        RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG     RESET SMS CHOISE
                BRA             ACTION10        JUMP TO END

ACTION10_10     CMP.B           #10,SEND_MSG    CMP NR OF CHOSEN SMS
                BNE             ACTION10_11     JUMP TO NEXT IF NOT THIS
                BSR             ERTEMCON        RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG     RESET SMS CHOISE
                BRA             ACTION10        JUMP TO END

ACTION10_11     CMP.B           #11,SEND_MSG    CMP NR OF CHOSEN SMS
                BNE             ACTION10_12     JUMP TO NEXT IF NOT THIS
                BSR             ERRANGE         RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG     RESET SMS CHOISE
                BRA             ACTION10        JUMP TO END

ACTION10_12     CMP.B           #12,SEND_MSG    CMP NR OF CHOSEN SMS
                BNE             ACTION10_13     JUMP TO NEXT IF NOT THIS
```

```
                BSR             ERALARAN            RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG         RESET SMS CHOISE
                BRA             ACTION10            JUMP TO END

ACTION10_13     CMP.B           #13,SEND_MSG        CMP NR OF CHOSEN SMS
                BNE             ACTION10_14         JUMP TO NEXT IF NOT THIS
                BSR             ERALACON            RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG         RESET SMS CHOISE
                BRA             ACTION10            JUMP TO END

ACTION10_14     CMP.B           #14,SEND_MSG        CMP NR OF CHOSEN SMS
                BNE             ACTION10_15         JUMP TO END IF NOT THIS
                BSR             ALARSTAT            RUN SMS SUBROUTINE
                MOVE.B          #0,SEND_MSG         RESET SMS CHOISE
                BRA             ACTION10            JUMP TO END

ACTION10_15     RTS                                 RETURN
***********************************************
ACTION12        NOP                                 ALARM OFF
                MOVE.B          #4,ALARM_COUNT1     EDETING THE ALARM COUNT
                MOVE.B          #2,SEND_MSG         CHOSING RESPOND SMS
                BSR             SEND_SMS            SENDING SMS
                RTS                                 RETURN
***********************************************
ACTION14        NOP                                 CONTROLE OFF
                BCLR.L          #0,D4               CLEAR CONTROLE STATUS BIT
                MOVE.B          #4,SEND_MSG         CHOSING RESPOND SMS
                BSR             SEND_SMS            SENDING SMS
                RTS                                 RETURN
***********************************************
ACTION15        NOP                                 STATUS
                MOVE.B          #5,SEND_MSG         CHOSING RESPOND SMS
                BSR             SEND_SMS            SENDING SMS
                RTS                                 RETURN
***********************************************
ACTION16        NOP                                 TEMP:
                LEA             TEMPX,A0            LOAD POINTER TO TEMPRATURE
                MOVE.L          #0,D0               RESET REGISTER
                MOVE.L          #0,D1               RESET REGISTER
ACTION16_03     CMP.L           A3,A5               CMP POINTER TO SEE IF AT END OF BUFFER
                BEQ             ACTION16_05         IF AT END JUMP

                CMP.B           #$30,(A3)           LOOK FOR ASCI NUMBER
                BLT             ACTION16_02         JUMP IF NOT MUNBER
                CMP.B           #$39,(A3)           *
                BGT             ACTION16_02         *

                ROL.L           #8,D1               ROTATE REGISTER TO FIT NUMBER
                ADD.L           #1,D0               COUNT NUMBERS
                MOVE.B          (A3),D1             MOVE NUMBER FROM BUFER TO REGSTER

                CMP.L           #1,D0               LOOK IF IT IS FIRST NUMBER
                BNE             ACTION16_09         JUMP IF ITS NOT

                MOVE.B          (A3),D2             MOVE ASCII TO REGISTER
                SUB.B           #$30,D2             CONVERT FROM ASCII
                MULU            #10,D2              MULTIPLY BY 10 TO GET THE RIGHT AMOUNT
ACTION16_09     CMP.L           #2,D0               LOOK IF ITS SECOND NUMBER
                BNE             ACTION16_10         JUMP IF NOT
                ADD.B           (A3),D2             ADD THE VALUE TO REGISTER
                SUB.B           #$30,D2             CONVERT FROM ACSII

ACTION16_10     CMP.B           #2,D0               LOOK IF THERE IS 2 NUMBERS
                BEQ             ACTION16_08         JUMP IF
```

```
ACTION16_02   ADD          #1,A3            MOVE POINTER
              BRA          ACTION16_03      JUMP

ACTION16_08   LEA          MAX_TEMP_A,A0    LOAD MAX TEMP TO REGISTER
              CMP.W        (A0),D1          CMP TEMPS
              BGE          ACTION16_06      IF GREATER THEN MAX JUMP

              LEA          MIN_TEMP_A,A0    LOAD MIN TEMP TO REGISTER
              CMP.W        (A0),D1          CMP TEMPS
              BLE          ACTION16_06      JUMP IF LESS THEN MIN TEMP

              CMP.W        #$3030,D1        LOOK IF TEMP IS INSIDE RANGE
              BLT          ACTION16_07      JUMP IF UNDER
              CMP.W        #$3430,D1        *
              BGT          ACTION16_07      JUMP IF OVER

              MOVE.W       D1,TEMPX         SAVE TEMP IN VARIABEL
              MOVE.B       D2,TEMPX_D       SAVE ASCII IN VARIBEL



ACTION16_04   BSET.L       #0,D4            SET CONTROLE STATUS BIT
              MOVE.B       #7,SEND_MSG      SELECT TEMPRATUR SET SMS
              BSR          SEND_SMS         SEND SMS
              RTS                           RETURN

ACTION16_05   MOVE.B       #8,SEND_MSG      SELECT ERROR SYNAX SMS
              BSR          SEND_SMS         SEND SMS
              RTS

ACTION16_06   MOVE.B       #10,SEND_MSG     SELECT CONFLICK WITH ALARM SMS
              BSR          SEND_SMS         SEND SMS
              RTS

ACTION16_07   MOVE.B       #11,SEND_MSG     SELECT TEMP OUTSIDE RNGE SMS
              BSR          SEND_SMS         SEND SMS
              RTS
***********************************************
ACTION17      NOP                           ALARM:
              MOVE.L       #0,D0            RESET REGISTER
              MOVE.L       #0,D1            RESET REGISTER
              MOVE.L       #0,D2            RESET REGISTER
              BTST.L       #23,D4           COTROL IF ADMINISTRATOR
              BEQ          ACTION17_10      JUMP IF NOT
ACTION17_03   CMP.L        A3,A5            LOOK IF AT END OF BUFFER
              BEQ          ACTION17_05      JUMP IF

              CMP.B        #$30,(A3)        CMP TO FIND ASCII NUMBER
              BLT          ACTION17_02      JUMP IF NOT
              CMP.B        #$39,(A3)        *
              BGT          ACTION17_02      *

              ROL.L        #8,D1            ROTATE REGISTER TO PLACE NUMBER RIGHT
              MOVE.B       (A3),D1          MOVE NUMBER TO REGISTER
              ADD.L        #1,D0            COUNT NUMBERS

              CMP.B        #4,D0            CMP IF AMOUNT OF NUMBER IS RIGHT
              BEQ          ACTION17_07      JUMP IF

ACTION17_02   ADD          #1,A3            MOVE POINTER
              BRA          ACTION17_03      JUMP

ACTION17_07   MOVE.W       D1,D2            SPLIT UP MIN AND MAX TEMP
```

```
                SWAP.W          D1              SWAP REGISTER TO FIT THE NUMBER PLACEMENT

                CMP.W           D1,D2           CMP MIN TO MAX TEMP
                BLE             ACTION17_05     IF MAX SMALLER THEN MIN JUMP

                CMP.W           #$3030,D1       CMP ALARM TEMP TO MIN LIMET
                BLT             ACTION17_08     JUMP IF UNDER
                CMP.W           #$3830,D2       CMP ALARM TEMP TO MAX
                BGT             ACTION17_08     JUMP IF OVER

                CMP.W           TEMPX,D1        CMP MIN ALARM TO SET TEMP
                BGE             ACTION17_09     JUMP IF GREATER
                CMP.W           TEMPX,D2        CMP MAX ALARM TO SET TEMP
                BLE             ACTION17_09     JUMP IF UNDER

                MOVE.W          D1,MIN_TEMP_A   SAVE ALARM MIN TEMP
                MOVE.W          D2,MAX_TEMP_A   SAVE ALARM MAX TEMP

                SUB.B           #$30,D1         CONVERT FROM ASCII
                MOVE.B          D1,MIN_TEMP     MOVE TO MIN VARIABEL
                ROR.L           #8,D1           ROTATE REGISTER
                SUB.B           #$30,D1         CONVERT FROM ASCII
                MULU            10,D1           MULTIPLY TO FIT
                ADD.B           D1,MIN_TEMP     ADD TO MIN VARIABEL

                SUB.B           #$30,D2         CONVERT FORM ASCII
                MOVE.B          D2,MAX_TEMP     MOVE TO MAX VARIABEL
                ROR.L           #8,D2           ROTATE REGISTER
                SUB.B           #$30,D2         CONVERT FROM ASCII
                MULU            10,D2           MULTIPLY TO FIL VALUE
                ADD.B           D2,MAX_TEMP     ADD TO MAX VARIABLE


ACTION17_04     MOVE.B          #9,SEND_MSG     SELECT ALARM SET SMS
                BSR             SEND_SMS        SENDING SMS
                RTS                             RETURN

ACTION17_05     MOVE.B          #8,SEND_MSG     SELECT ERROR SYNTAX SMS
                BSR             SEND_SMS        SENDING SMS
                RTS                             RETURN

ACTION17_08     MOVE.B          #12,SEND_MSG    SELECT ERROR ALARM RANGE
                BSR             SEND_SMS        SENDING SMS
                RTS                             RETURN

ACTION17_09     MOVE.B          #13,SEND_MSG    SELECT ALARM CONFLICT WITH SET TEMP
                BSR             SEND_SMS        SENDING SMS
                RTS                             RETURN

ACTION17_10     MOVE.B          #6,SEND_MSG     SELECT NOT ALLOWED SMS
                BSR             SEND_SMS        SENDING SMS
                RTS                             RETURN
*************************************************
ACTION18        LEA             HANG_UP,A0      LOAD COMMAND POINTER
                BSR             SEND_ST_GSM     SENDING STRING
                RTS                             RETURN
*************************************************
MOVE_BUF        NOP                             RESETS THE BUFFER
                LEA             BUF_S_POINT,A4  SET BUFFER START POINTER TO START OF BUFFER SPACE
                CMP.L           A4,A5           CMP TO SE IF END OF BUFFER
                BEQ             MOVE_BUF1       JUMP IF
                MOVE.B          #0,-(A5)        RESET SPACE AND MOVE POINTER
                BRA             MOVE_BUF        JUMP
MOVE_BUF1       MOVE.B          #0,INCOUNTER    RESET THE LF COUNTER
```

```
                RTS                                     RETURN
*************************************************
SEND_TE_GSM     NOP                                     SEND TEMPRATURE TO SMS
                ROL.L           #8,D7                   ROTATE TEMP REGISTER TO GET FIRST CHAR
SEND_TE_GSM_1   BTST.B          #1,GSM_SR               SE IF GSM READY
                BEQ             SEND_TE_GSM_1           IF NOT JUMP
                MOVE.B          D7,GSM_DR               MOVE CHAR TO SMS
                ROL.L           #8,D7                   RETATE REGISTER TO GET NEXT CHAR
SEND_TE_GSM_2   BTST.B          #1,GSM_SR               SE IF GSM READY
                BEQ             SEND_TE_GSM_2           IF NOT JUMP
                MOVE.B          D7,GSM_DR               MOVE CHAR TO SMS
                ROL.L           #8,D7                   RETATE REGISTER TO GET NEXT CHAR
SEND_TE_GSM_3   BTST.B          #1,GSM_SR               SE IF GSM READY
                BEQ             SEND_TE_GSM_3           IF NOT JUMP
                MOVE.B          D7,GSM_DR               MOVE CHAR TO SMS
SEND_TE_GSM_4   BTST.B          #1,GSM_SR               SE IF GSM READY
                BEQ             SEND_TE_GSM_4           IF NOT JUMP
                MOVE.B          #',',GSM_DR             SEND STATIC CHAR
                ROL.L           #8,D7                   RETATE REGISTER TO GET NEXT CHAR
SEND_TE_GSM_5   BTST.B          #1,GSM_SR               SE IF GSM READY
                BEQ             SEND_TE_GSM_5           IF NOT JUMP
                MOVE.B          D7,GSM_DR               MOVE CHAR TO SMS
                RTS                                     RETURN
*************************************************
SEND_SMS_A      NOP                                     SEND SMS FOR ADMIN
                BSR             MOVE_BUF                RESET BUFFER
                LEA             SEND,A0                 LOAD COMMAND
                BSR             SEND_ST_GSM             SEND COMMAND
                LEA             ADMIN_NR2,A0            *
                BSR             SEND_ST_GSM             *
                LEA             COM_END,A0              *
                BSR             SEND_ST_GSM             *
                BSR             WAIT1LF                 WAIT FOR 1 LINE FEED
                BSR             CMP_ANS                 RECONISE RESPONSE
                BSR             WAIT2LF                 WAI 2 LINE FEED
                BSR             CMP_ANS                 RECONISE RESPONSE
                BSR             MOVE_BUF                RESET BUFFER
                RTS                                     RETURN
*************************************************
```

H.3 Constants

```
* CODES FOR GSM MODULE
GSM_SR          EQU             $B00001         GSM STATUS AND CONTROLE REGISTER
GSM_DR          EQU             $B00003         GSM DATA REGISTER
GSM_DS          EQU             $50000          DATA SPACE
BUF_S_POINT     EQU             $47000
*****************************************

*****************************************
****START OF THE GSM COMMAND DATA SPACE*****
*****************************************ACIA SETUP
ACIA_I          DC.B            $D5             %11010101
*****************************************PREDEFINED COMMANDS
E0              DC.B            'ATE0',13,0                     ECHO OFF
CNMI            DC.B            'AT+CNMI=2,1,0,0,1',13,0        SET INTERRUPTE
CMGF            DC.B            'AT+CMGF=1',13,0                SET TEXT MODE
CSQ             DC.B            'AT+CSQ',13,0                   GSM NETWORK STATUS
CCLK_Q          DC.B            'AT+CCLK?',13,0                 GET THE TIME
SEND            DC.B            'AT+CMGS=',0                    SEND SMS
DELET           DC.B            'AT+CMGD=',0                    DELET SMS
READ            DC.B            'AT+CMGR=',0                    READ SMS
```

```
COM_END     DC.B    13,0                        END CHAR FOR AT COMMANDS
HANG_UP     DC.B    'AT+CHCP',0                 HANG UP A CALL
*****************************************SET CLOCK COMMAND
CCLK        DC.B    'AT+CCLK="'
YY          DC.B    '0','9'
            DC.B    '/'
MM          DS.B    2
            DC.B    '/'
DD          DS.B    2
            DC.B    ','
HH          DS.B    2
            DC.B    ':'
MI          DS.B    2
            DC.B    ':'
SE          DC.B    '0','0'
            DC.B    '"',13,0
*****************************************SET PIN COMMAND
CPIN        DC.B    'AT+CPIN="'
PIN         DS.B    4
            DC.B    '"',13,0
*****************************************LINKED LIST
ELM0        DC.L    DATA0,ACTION0,ELM1
ELM1        DC.L    DATA1,ACTION0,ELM2
ELM2        DC.L    DATA2,ACTION0,ELM3
ELM3        DC.L    DATA3,ACTION3,ELM4
ELM4        DC.L    DATA4,ACTION4,ELM5
ELM5        DC.L    DATA5,ACTION5,ELM6
ELM6        DC.L    DATA6,READ_MSG,ELM7
ELM7        DC.L    DATA7,ACTION7,ELM8
ELM8        DC.L    DATA8,ACTION8,ELM9
ELM9        DC.L    DATA9,ACTION9,ELM10
ELM10       DC.L    DATA10,ACTION10,ELM12
ELM12       DC.L    DATA12,ACTION12,ELM14
ELM14       DC.L    DATA14,ACTION14,ELM15
ELM15       DC.L    DATA15,ACTION15,ELM16
ELM16       DC.L    DATA16,ACTION16,ELM17
ELM17       DC.L    DATA17,ACTION17,ELM18
ELM18       DC.L    DATA18,ACTION18,0
*********************************************DATA STRINGS FOR LINKED LIST
DATA0       DC.B    'ATE0',13,13,10,'OK',13,10,0        OK RESPONSE WITH ECHO
DATA1       DC.B    13,10,'OK',13,10,0                  OK RESPONSE
DATA2       DC.B    13,10,'+CME␣ERROR:␣operation␣not␣allowed',13,10,0 TEST STRING
DATA3       DC.B    13,10,'+CME␣ERROR:␣incorrect␣password',0          WRONG PIN CODE
DATA4       DC.B    13,10,'+CME␣ERROR:␣invalid␣index',0               WRONG TIME
DATA5       DC.B    13,10,'+CMS␣ERROR:␣SIM␣busy',0                    SIM CARD NOT READY
DATA6       DC.B    13,10,'+CMTI:␣"SM",',0                            SMS RECIVED
DATA7       DC.B    13,10,'+CMGS:',0                                  SMS SENT
DATA8       DC.B    13,10,'+CSQ:␣',0                                  NETWORK STRENGT
DATA9       DC.B    13,10,'+CCLK:␣',0                                 THE TIME
DATA10      DC.B    $0D,$0A,$3E,$20,0                                 TRANSMIT SMS
DATA12      DC.B    'alarm␣off',0                                     STOP THE ALARM SMS
DATA14      DC.B    'stop',0                                          TURN OFF CONTROLE
DATA15      DC.B    'status',0                                        STATUS
DATA16      DC.B    'settemp',0                                       SETS THE TEMP
DATA17      DC.B    'setalarm',0                                      SETS THE ALARM TEMP
DATA18      DC.B    13,10,'RING',0                                    INCOMMING CALLL
*******************************************
****START OF THE GSM VARIABEL SPACE*********
*******************************************
MSG_NR      DS.B    1           IDENTIFYING RECIVED MESSAGES
            DC.B    0
SEND_MSG    DS.B    1           IDENTIFYING MESSAGES TO SEND
            DC.B    0
***************************************
```

```
TEMP_NR         DS.B            13          NUMBER OF THE CURRENT REMOTE USER
                DC.B            0
************************************ COUNTERS FOR BALANCING SYSTEM UPDATES
INCOUNTER       DS.B            1           LINE FEED COUNTER
CLKCOUNTER      DS.B            1           CLOCK COUNTER
GSMCOUNTER      DS.B            1           GSM COUNTER
TEMPCOUNTER     DS.B            1           TEMPRATURE COUNTER
ALARMCOUNTER    DS.B            1           ALARM COUNTER
```

Appendix I

LCD software

I.1 Main Functions

```
*GSM RCS 2008
*****************************************
TIME_OUT        EQU         500000          THE TIMEOUT IS APPROX 20s

*SPACE
*****************************************
ALARM_COUNT2    DS.L        1               THE COUNTER USED WHEN SENDING ALARM TO ADMINISTRATOR
ALARM_COUNT1    DS.B        1               THE MESSAGE COUNTER USED WHEN SENDING ALARM TO ADMINISTRATOR
MIN_TEMPOLD     DS.B        1               THE SET MIN TEMPERATURE OLD
MAX_TEMPOLD     DS.B        1               THE SET MAX TEMPERATURE OLD
TEMP_OLD        DS.B        1               THE SET TEMPERATURE OLD
TEMP_OLD_A      DS.W        1
MIN_TEMP        DS.B        1               THE SET MIN TEMPERATURE (ALARM)
MAX_TEMP        DS.B        1               THE SET MAX TEMPERATURE (ALARM)
MIN_TEMP_A      DS.W        1               THE SET MIN TEMPERATURE (ALARM)
                DC.B        0
MAX_TEMP_A      DS.W        1               THE SET MAX TEMPERATURE (ALARM)
                DC.B        0
ADMIN_NR1       DS.B        8               THE ADMIN PHONE NUMBER (USED AS ACESS CODE)
ADMIN_NR2       DC.B        '"+45'
ADMIN_NR        DS.B        8               THE ADMIN PHONE NUMBER (NOTE THAT THIS IS THE "REAL" NUMBER )
                DC.B        '"',0
STOP1           DC.B        0
USER_NR2        DC.B        '"+45'
USER_NR         DS.B        8               THE USER PHONE NUMBER
                DC.B        '"',0
STOP2           DC.B        0
DAYS_ACTIVE     DS.B        1               USER, DAYS ACTIVE
DAYS_ACTIVE_C   DS.B        1               USER, DAYS ACTIVE    NOTE NOT ACSII
TEMPX           DS.W        1               THE SET TEMPERATURE IN ACCII
                DC.B        0
TEMPX_D         DS.B        1               THE SET TEMPERATURE IN DECIMAL
ENTRY           DS.B        1               THE ENTRY FOR THE KEYPAD
DAYS_COUNT      DS.B        1
TEMP_C_D        DS.B        1
TEMP_AV_D       DS.W        1


*CONSTANTS
OFFLINE         DC.B        'Offline',0
```

```
ONLINE          DC.B        ' Online',0
WRONG_TIME      DC.B        'Wrong time/date',0
WRONG_PIN       DC.B        'Wrong PIN-code',0
RESETUSER0      DC.B        'Reset user',0
CONFIRM         DC.B        'Confirm       1',0
CTEMP           DC.B        'C. Temp.',0
CONTROL         DC.B        'Control',0
TIME            DC.B        'Time',0
GSM             DC.B        'GSM',0
TIME2           DC.B        'Time/Date',0
TIME3           DC.B        '  :     /  ',0
USER_MODE       DC.B        'User mode',0
NEW_USER1       DC.B        'User          1',0
NEW_USER        DC.B        'User',0
RESET_USER      DC.B        'Reset user     2',0
ADMIN           DC.B        'Administrator',0
ADMIN3          DC.B        'Administrator  3',0
ENTER           DC.B        'Enter',0
PHONE_NR1       DC.B        'Phone number',0
PHONE_NR2       DC.B        '+45',0
DAYSACTIVE      DC.B        'Days active',0
SET_TEMPM       DC.B        'Set Temperature',0
CONTROL2        DC.B        'Control       2',0
ALARM           DC.B        'Alarm         3',0
BACK_1          DC.B        'BACK          1',0
BACK_2          DC.B        'BACK          2',0
BACK_3          DC.B        'BACK          3',0
CURRENT         DC.B        'Current',0
TEMP_LONG       DC.B        'Temperature    1',0
TEMP_NULL       DC.B        'Temperature',0
TEMP_RANGE      DC.B        'XX-XX',167,'C',0
USER_TOP        DC.B        'User',0
TURN_ON         DC.B        'Turn ON       1',0
TURN_OFF        DC.B        'Turn OFF      2',0
ALARMM          DC.B        'Alarm',0
SET_ALARM       DC.B        'Set           1',0
ALARM_TEMP      DC.B        'Alarm temp',0
PIN_REG         DC.B        'PIN code ',0
GSM_RCS         DC.B        '     GSM RCS    ',0
ES_DEP          DC.B        '  ES Department ',0
AAU2008         DC.B        '     AAU 2008   ',0
STATE_ON        DC.B        ' ON',0
STATE_OFF       DC.B        'OFF',0

*****************************************
*****************************************

*MENU FUNCTIONS
*************************************************************************************************************************
*THE MAIN RUTINE FOR THE MENU SYSTEM
*****************************************
MENUMAIN        NOP
MENU_1          CMP.L       #1,D3
                BNE         MENU_2
                BSR         P_INIITIALIZE    FIRST MENU DISPLAYED DURING INITIALIZATION
                BRA         ENDMENU
MENU_2          CMP.L       #2,D3
                BNE         MENU_3
                BSR         P_ADMINREG1      ENTER ADMINISTRATOR PHONE NUMBER
                BRA         ENDMENU
MENU_3          CMP.L       #3,D3
                BNE         MENU_4
                BSR         P_ADMINREG2      ENTER GSM - PIN CODE
                BRA         ENDMENU
```

```
MENU_4      CMP.L     #4,D3
            BNE       MENU_5
            BSR       P_ADMINREG3     ENTER TIME AND DATE
            BRA       ENDMENU
MENU_5      CMP.L     #5,D3
            BNE       MENU_6
            BSR       P_MODE          THE USER MODE MENU
            BRA       ENDMENU
MENU_6      CMP.L     #6,D3
            BNE       MENU_7
            BSR       P_RESET_U       THE USER PROFILE RESET MENU
            BRA       ENDMENU
MENU_7      CMP.L     #7,D3
            BNE       MENU_8
            BSR       P_ADMINAUTH     THE ADMINISTRATOR AUTHENTICATION MENU
            BRA       ENDMENU
MENU_8      CMP.L     #8,D3
            BNE       MENU_9
            BSR       P_ADMINMAIN     THE ADMINISTRATOR MAIN MENU
            BRA       ENDMENU
MENU_9      CMP.L     #9,D3
            BNE       MENU_10
            BSR       P_CONTROL_A     THE ADMINISTRATOR CONTROL MENU
            BRA       ENDMENU
MENU_10     CMP.L     #10,D3
            BNE       MENU_11
            BSR       P_ALARM_A       THE ADMINISTRATOR ALARM MENU
            BRA       ENDMENU
MENU_11     CMP.L     #11,D3
            BNE       MENU_12
            BSR       P_SETALARM_A    THE ADMINISTRATOR ENTER ALARM TEMPERATURE MENU
            BRA       ENDMENU
MENU_12     CMP.L     #12,D3
            BNE       MENU_13
            BSR       P_TEMP_A        THE ADMINISTRATOR TEMPERATURE MENU
            BRA       ENDMENU
MENU_13     CMP.L     #13,D3
            BNE       MENU_14
            BSR       P_SETTEMP_A     THE ADMINISTRATOR ENTER SET TEMPERATURE
            BRA       ENDMENU
MENU_14     CMP.L     #14,D3
            BNE       MENU_15
            BSR       P_USERREG1      THE USER ENTER PHONE NUMBER MENU (OMITTED WHEN A PROFILE IS ACTIVE)
            BRA       ENDMENU
MENU_15     CMP.L     #15,D3
            BNE       MENU_16
            BSR       P_USERREG_2     THE USER INTER DAYS ACTIVE MENU (OMITTED WHEN A PROFILE IS ACTIVE)
            BRA       ENDMENU
MENU_16     CMP.L     #16,D3
            BNE       MENU_17
            BSR       P_USERMAIN      THE USER MAIN MENU
            BRA       ENDMENU
MENU_17     CMP.L     #17,D3
            BNE       MENU_18
            BSR       P_CONTROL_U     THE USER CONTROL MENU
            BRA       ENDMENU
MENU_18     CMP.L     #18,D3
            BNE       MENU_19
            BSR       P_TEMP_U        THE USER TEMPERATURE MENU
            BRA       ENDMENU
MENU_19     CMP.L     #19,D3
            BNE       MENU_20
            BSR       P_SETTEMP_U     THE USER ENTER SET TEMPERATURE MENU
            BRA       ENDMENU
```

```
MENU_20         CMP.L       #20,D3
                BNE         ENDMENU
                BSR         P_STATUS        THE STATUS MENU
                BRA         ENDMENU
ENDMENU         RTS                         TERMINATE THE RUTINE AND RETURN TO NORMAL


* THE MENUS USED IN THE MAINMENU PROGRAM

*1 ADMINISTRATOR REGISTRATION               THE MENU DISPLAYED DURING RESET/SETUP
* FINISHED
**************************************
P_INIITIALIZE MOVE.B      #$C,$900001       DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                BSR         RESET_LCD
                LEA         GSM_RCS,A6
                BSR         PRINT_LINE
                BSR         SECOND
                LEA         ES_DEP,A6
                BSR         PRINT_LINE
                BSR         THIRD
                LEA         AAU2008,A6
                BSR         PRINT_LINE
                MOVE.L      #5,D3
                RTS
**************************************END

*2 ADMINISTRATOR REGISTRATION PHONE NUMBER
*FINISHED
**************************************
P_ADMINREG1     MOVE.B      #$F,$900001       DISPLAY ON AND CURSOR ON
                BSR         WAIT              MENU SETUP
                BSR         RESET_LCD         Administrator
                LEA         ADMIN,A6          Enter
                BSR         PRINT_LINE        Phone number
                BSR         SECOND            45+
                LEA         ENTER,A6
                BSR         PRINT_LINE
                BSR         THIRD
                LEA         PHONE_NR1,A6
                BSR         PRINT_LINE
                BSR         FOURTH
                LEA         PHONE_NR2,A6
                BSR         PRINT_LINE
**************************************ENTRY
                BSR         WAIT
                MOVE.L      #0,D2
                LEA         ADMIN_NR,A0       A0 POINTS TO THE ADMIN PHONE NUMBER SPACE
RUNAGIN16       BSR         KEYPAD_C          THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BEQ         RUNAGIN16
ACTIVE16        CMP.L       #'A',D5
                BEQ         RUNAGIN16         END IF THE USER PRESSES * AND 8 ENTRY HAS BEEN MADE
                ADD.L       #1,D2
                MOVE.B      D5,$900003        PLACE THE ENTRY ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      D5,(A0)+          SAVE THE INPUT AS ACSII IN THE ADMIN_NR1 SPACE
                CMP.L       #8,D2
                BEQ         END16
                BRA         RUNAGIN16
END16           MOVE.B      #$C,$900001       DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                BSR         WAIT_OUT
                MOVE.L      #3,D3
```

```
                RTS
**************************************END

*3 ADMINISTRATOR REGISTRATION GSM - PINCODE
*FINISHED
**************************************
P_ADMINREG2     MOVE.B      #$F,$900001     DISPLAY ON AND CURSOR ON
                BSR         WAIT
                BSR         RESET_LCD       Administrator
                LEA         ADMIN,A6        Enter
                BSR         PRINT_LINE      PIN code
                BSR         SECOND
                LEA         ENTER,A6
                BSR         PRINT_LINE
                BSR         THIRD
                LEA         PIN_REG,A6
                BSR         PRINT_LINE
                BSR         FOURTH
**************************************ENTRY
                BSR         WAIT
                MOVE.L      #0,D2
                LEA         PIN,A0
RUNAGIN20       BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BEQ         RUNAGIN20
ACTIVE20        CMP.L       #'A',D5
                BEQ         RUNAGIN20
                MOVE.B      D5,$900003      PLACE THE ENTRY ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      D5,(A0)+        SAVE THE INPUT
                ADD.L       #1,D2
                CMP.L       #4,D2
                BEQ         END20
                BRA         RUNAGIN20
END20           MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                BSR         WAIT_OUT
                MOVE.L      #4,D3
                RTS
**************************************END

*21 ADMINISTRATOR REGISTRATION WRONG  GSM - PINCODE
**************************************
P_ADMINREG4     MOVE.B      #$F,$900001     DISPLAY ON AND CURSOR ON
                BSR         WAIT
                BSR         RESET_LCD       Wrong PIN-code
                LEA         WRONG_PIN,A6    Enter
                BSR         PRINT_LINE      PIN code
                BSR         SECOND
                LEA         ENTER,A6
                BSR         PRINT_LINE
                BSR         THIRD
                LEA         PIN_REG,A6
                BSR         PRINT_LINE
                BSR         FOURTH
**************************************ENTRY
                BSR         WAIT
                MOVE.L      #0,D2
                LEA         PIN,A0
RUNAGIN25       BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BEQ         RUNAGIN25
ACTIVE25        CMP.L       #'A',D5
                BEQ         RUNAGIN25
```

```
                MOVE.B      D5,$900003      PLACE THE ENTRY ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      D5,(A0)+        SAVE THE INPUT
                ADD.L       #1,D2
                CMP.L       #4,D2
                BEQ         END25
                BRA         RUNAGIN25
END25           MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                BSR         WAIT_OUT
                MOVE.L      #5,D3
                RTS
*************************************END

*22 ADMINISTRATOR REGISTRATION TIME AND DATE
*FINISHED
*************************************
P_ADMINREG5     MOVE.B      #$F,$900001     DISPLAY ON AND CURSOR ON
                BSR         WAIT
                BSR         RESET_LCD
                LEA         WRONG_TIME,A6
                BSR         PRINT_LINE
                BSR         SECOND
                LEA         ENTER,A6
                BSR         PRINT_LINE
                BSR         THIRD
                LEA         TIME2,A6
                BSR         PRINT_LINE
                BSR         FOURTH
                LEA         TIME3,A6
                BSR         PRINT_LINE
                MOVE.B      #$D0,$900001    LOCATION COMMAND FOURTH LINE
*************************************ENTRY
                BRA         ENTERTIME
                RTS

*4 ADMINISTRATOR REGISTRATION TIME AND DATE
*FINISHED
*************************************
P_ADMINREG3     MOVE.B      #$F,$900001     DISPLAY ON AND CURSOR ON
                BSR         WAIT
                BSR         RESET_LCD
                LEA         ADMIN,A6
                BSR         PRINT_LINE
                BSR         SECOND
                LEA         ENTER,A6
                BSR         PRINT_LINE
                BSR         THIRD
                LEA         TIME2,A6
                BSR         PRINT_LINE
                BSR         FOURTH
                LEA         TIME3,A6
                BSR         PRINT_LINE
                MOVE.B      #$D0,$900001    LOCATION COMMAND FOURTH LINE
*************************************ENTRY
ENTERTIME       MOVE.L      #0,D2
RUNAGIN21       BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BEQ         RUNAGIN21
ACTIVE21        CMP.L       #'A',D5
                BEQ         RUNAGIN21
                MOVE.B      D5,$900003      PLACE THE ENTRY ON THE DISPLAY
                BSR         WAITBF
                ADD.L       #1,D2
```

```
                BSR         GSMTIME
                MOVE.B      D5,(A0)+
                CMP.L       #2,D2
                BEQ         BREAK1
                CMP.L       #4,D2
                BEQ         BREAK2
                CMP.L       #6,D2
                BEQ         BREAK3
                CMP.L       #8,D2
                BEQ         END21
                BRA         RUNAGIN21
BREAK1          MOVE.B      #$3A,$900003    WRITE : ON DISPLAY
                BRA         RUNAGIN21
BREAK2          MOVE.B      #$20,$900003    WRITE (SPACE)  ON DISPLAY
                BRA         RUNAGIN21
BREAK3          MOVE.B      #$2F,$900003    WRITE / ON DISPLAY
                BRA         RUNAGIN21
END21           MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                BSR         WAIT_OUT
                MOVE.L      #1,D3           NEXT MENU THE INITIALIZEION MENU
                RTS

GSMTIME         CMP.L       #1,D2
                BNE         GSMTI_MI
                LEA         HH,A0

GSMTI_MI        CMP.L       #3,D2
                BNE         GSMTI_MM
                LEA         MI,A0

GSMTI_MM        CMP.L       #5,D2
                BNE         GSMTI_DD
                LEA         MM,A0

GSMTI_DD        CMP.L       #7,D2
                BNE         GSMTI_END
                LEA         DD,A0

GSMTI_END       RTS
**************************************END

*5  JUMP MENU "MODE_MENU"
*FINISHED
**************************************
P_MODE          MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT            MENU SETUP
                BSR         RESET_LCD       *****************
                LEA         USER_MODE,A6    *User mode       *
                BSR         PRINT_LINE      *New user       1*
                BSR         SECOND          *Reset user     2*
                LEA         NEW_USER1,A6    *Administrator 3*
                BSR         PRINT_LINE      *****************
                BSR         THIRD
                LEA         RESET_USER,A6
                BSR         PRINT_LINE
                BSR         FOURTH
                LEA         ADMIN3,A6
                BSR         PRINT_LINE
**************************************
RUNAGIN1        MOVE.L      #0,D0           THE COUNTER IS RESET
L1              BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BNE         ACTIVE_1
```

```
                ADD.L       #1,D0           TIMEOUT COUNTER INCREMENTED
                CMP.L       #TIME_OUT,D0    TIME OUT IF D0 = 30
                BEQ         TIMEOUT1
                BRA         L1
ACTIVE_1        CMP.L       #'1',D5
                BNE         LOOP1_2
                MOVE.L      #14,D3          1. USER ENTER PHONE NUMBER
                BRA         END1
LOOP1_2         CMP.L       #'2',D5
                BNE         LOOP1_3
                MOVE.L      #6,D3           2. RESET USER MENU
                BRA         END1
LOOP1_3         CMP.L       #'3',D5
                BNE         RUNAGIN1
                MOVE.L      #7,D3           3. ADMINISTRATOR AUTHENTICATION
                BRA         END1
TIMEOUT1        MOVE.L      #20,D3
                BCLR        #2,D4           SETS THE THIRD BIT IN D4 LOW
END1            RTS
*************************************END

*6 JUMP MENU "RESET_USER"
*FINISHED
***************************************
P_RESET_U       MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT            MENU SETUP
                BSR         RESET_LCD       ****************
                LEA         RESETUSER0,A6   *Reset user     *
                BSR         PRINT_LINE      *Confirm       1*
                BSR         SECOND          *Back          2*
                LEA         CONFIRM,A6      *               *
                BSR         PRINT_LINE      ****************
                BSR         THIRD
                LEA         BACK_2,A6
                BSR         PRINT_LINE
***************************************
RUNAGIN12       MOVE.L      #0,D0           THE COUNTER IS RESET
L122            BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BNE         ACTIVE_12
                ADD.L       #1,D0           TIMEOUT COUNTER INCREMENTED
                CMP.L       #TIME_OUT,D0    TIME OUT IF D0 = 30
                BEQ         TIMEOUT12
                BRA         L122
ACTIVE_12       CMP.L       #'1',D5
                BNE         LOOP12_2
                MOVE.L      #5,D3           1. RESET THE USER PROFILE AND RETURN TO USER MODE MENU
                MOVE.L      #0,USER_NR
                MOVE.B      #0,DAYS_ACTIVE
                BRA         END12
LOOP12_2        CMP.L       #'2',D5
                BNE         RUNAGIN12
                MOVE.L      #5,D3           2. USER MODE MENU
                BRA         END12
TIMEOUT12       MOVE.L      #20,D3          RETURN TO STATUS
                BCLR        #2,D4           SETS THE THIRD BIT IN D4 LOW
END12           RTS
*************************************END

*7 (8) NUMERIC MENU "ADMIN_PHONENUMBER"
*FINISHED
***************************************
P_ADMINAUTH     MOVE.B      #$F,$900001     DISPLAY ON AND CURSOR ON
                BSR         WAIT            MENU SETUP
```

```
                BSR         RESET_LCD       ****************
                LEA         ADMIN,A6        *Administrator *
                BSR         PRINT_LINE      *Enter         *
                BSR         SECOND          *Phone number  *
                LEA         ENTER,A6        *+45           *
                BSR         PRINT_LINE      ****************
                BSR         THIRD
                LEA         PHONE_NR1,A6
                BSR         PRINT_LINE
                BSR         FOURTH
                LEA         PHONE_NR2,A6
                BSR         PRINT_LINE
*************************************
                MOVE.B      #$D3,$900001    LOCATION COMMAND FOURTH LINE + 4
                BSR         WAIT
                LEA         ADMIN_NR1,A1    A1 POINTS TO THE ADMIN PHONE NUMBER SPACE
                MOVE.L      #0,D2           COUNTER RESET
RUNAGIN17       MOVE.L      #0,D0           COUNTER RESET
RUN17           BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BNE         ACTIVE17
                CMP.L       #TIME_OUT,D0    TIME OUT IF D0 = 30
                BEQ         TIMEOUT17
                BRA         RUN17
ACTIVE17        CMP.L       #'A',D5
                BEQ         RUNAGIN17
                MOVE.B      D5,$900003      PLACE THE ENTRY ON THE DISPLAY
                BSR         WAITBF
                ADD.L       #1,D2
                MOVE.B      D5,(A1)+        SAVE THE INPUT
                CMP.L       #8,D2
                BNE         RUNAGIN17
                LEA         ADMIN_NR,A0     A1 POINTS TO THE BEGINNING OF THE ADMIN PHONE NUMBER
                LEA         ADMIN_NR1,A1    A2 POINTS TO THE BEGINNING OF THE "REAL" ADMIN PHONE NUMBER
COM17           CMP.B       #0,(A0)
                BEQ         RIGHT
                MOVE.B      (A0)+,D0
                CMP.B       (A1)+,D0
                BNE         WRONG17
                BRA         COM17
RIGHT           MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                MOVE.L      #8,D3           IF THE PHONE NUMBER IS CORRECT GO TO ADMINISTRATOR MAIN MENU
                BSR         WAIT_OUT
                RTS
TIMEOUT17       MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                MOVE.L      #20,D3
                BCLR        #2,D4           SETS THE THIRD BIT IN D4 LOW
                RTS
WRONG17         BSR         WAIT_OUT
                MOVE.L      #5,D3           RETURN TO USER MODE MENU (PLACE ERROR MESSAGE BEFORE RETURN)
                RTS
*************************************END

*8 JUMP MENU "ADMIN_MAIN_MENU"
*FINISHED
*************************************
P_ADMINMAIN     MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT            MENU SETUP
                BSR         RESET_LCD       ****************
                LEA         ADMIN,A6        *Administrator *
                BSR         PRINT_LINE      *Temperature  1*
                BSR         SECOND          *Control      2*
```

```
                LEA         TEMP_LONG,A6    *Alarm          3*
                BSR         PRINT_LINE      ****************
                BSR         THIRD
                LEA         CONTROL2,A6
                BSR         PRINT_LINE
                BSR         FOURTH
                LEA         ALARM,A6
                BSR         PRINT_LINE
**************************************
RUNAGIN3        MOVE.L      #0,D0           THE COUNTER IS RESET
L3              BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BNE         ACTIVE_3
                ADD.L       #1,D0           TIMEOUT COUNTER INCREMENTED
                CMP.L       #TIME_OUT,D0    TIME OUT IF D0 = 30
                BEQ         TIMEOUT3
                BRA         L3
ACTIVE_3        CMP.L       #'1',D5
                BNE         LOOP3_2
                MOVE.L      #12,D3          1. TEMPERATURE MENU ADMINISTRATOR
                BRA         END3
LOOP3_2         CMP.L       #'2',D5
                BNE         LOOP3_3
                MOVE.L      #9,D3           2. CONTROL MENUS ADMINISTRATOR
                BRA         END3
LOOP3_3         CMP.L       #'3',D5
                BNE         RUNAGIN3
                MOVE.L      #10,D3          3. ALARM MENU ADMINISTRATOR
                BRA         END3
TIMEOUT3        MOVE.L      #20,D3
                BCLR        #2,D4           SETS THE THIRD BIT IN D4 LOW
END3            RTS
**************************************END

*9 JUMP MENU CONTROL MENU ADMIN
*FINISHED
**************************************
P_CONTROL_A     MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT            MENU SETUP
                BSR         RESET_LCD       ****************
                LEA         CONTROL,A6      *CONTROL         *
                BSR         PRINT_LINE      *Turn ON        1*
                BSR         SECOND          *Turn OFF       2*
                LEA         TURN_ON,A6      *Back           3*
                BSR         PRINT_LINE      ****************
                BSR         THIRD
                LEA         TURN_OFF,A6
                BSR         PRINT_LINE
                BSR         FOURTH
                LEA         BACK_3,A6
                BSR         PRINT_LINE
**************************************
RUNAGIN5        MOVE.L      #0,D0           THE COUNTER IS RESET
L5              BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BNE         ACTIVE_5
                ADD.L       #1,D0           TIMEOUT COUNTER INCREMENTED
                CMP.L       #TIME_OUT,D0    TIME OUT IF D0 = 30
                BEQ         TIMEOUT5
                BRA         L5
ACTIVE_5        CMP.L       #'1',D5         1. TURN ON
                BNE         LOOP5_2
                BSET        #0,D4           SETS THE FIRST BIT IN D4 HIGH I.E. THE SYSTEM IS IN CONTROL MODE
                BRA         END5
```

```
LOOP5_2         CMP.L       #'2',D5          2. TURN OFF
                BNE         LOOP5_3
                BCLR        #0,D4            SETS THE FIRST BIT IN D4 LOW  I.E. THE SYSTEM IS IN MONITORING MODE
                BRA         END5
LOOP5_3         CMP.L       #'3',D5          3. BACK
                BNE         RUNAGIN5
                BRA         END5
TIMEOUT5        MOVE.L      #20,D3           STATUS MENU
                BCLR        #2,D4            SETS THE THIRD BIT IN D4 LOW
                RTS
END5            MOVE.L      #8,D3            RETURN TO ADMINISTRATOR MAIN MENU
                RTS
*************************************END

*10 JUMP MENU "ALARM_MENU"
*ALMOST FINISHED
**************************************
P_ALARM_A       MOVE.B      #$C,$900001      DISPLAY ON AND CURSOR OFF
                BSR         WAIT             MENU SETUP
                BSR         RESET_LCD        ****************
                LEA         ALARMM,A6        *Alarm          *
                BSR         PRINT_LINE       *Current        *
                BSR         SECOND           *Set           1*
                LEA         CURRENT,A6       *Back          2*
                BSR         PRINT_LINE       ****************
                BSR         THIRD
                LEA         SET_ALARM,A6
                BSR         PRINT_LINE
                BSR         FOURTH
                LEA         BACK_2,A6
                BSR         PRINT_LINE
                *UPDATE THE VARIABLE
                MOVE.B      #$CB,$900001     LOCATION COMMAND SECOND LINE – 5
                BSR         WAIT
                LEA         MIN_TEMP_A,A6
                MOVE.B      (A6)+,$900003       PRINT THE FIRST DIGIT OF THE MIN TEMPERATURE ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      (A6),$900003     PRINT THE SECOND DIGIT OF THE MIN TEMPERATURE ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      #$2D,$900003     PRINT ( – ) ON THE DISPLAY
                BSR         WAITBF
                LEA         MAX_TEMP_A,A6
                MOVE.B      (A6)+,$900003       PRINT THE FISRT DIGIT OF THE MAX TEMPERATURE ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      (A6),$900003     PRINT THE SECOND DIGIT OF THE MAX TEMPERATURE ON THE DISPLAY
                BSR         WAITBF
**************************************
RUNAGIN4        MOVE.L      #0,D0  THE COUNTER IS RESET
L4              BSR         KEYPAD_C      THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BNE         ACTIVE_4
                ADD.L       #1,D0       TIMEOUT COUNTER INCREMENTED
                CMP.L       #TIME_OUT,D0     TIME OUT IF D0 = 30
                BEQ         TIMEOUT4
                BRA         L4
ACTIVE_4        CMP.L       #'1',D5
                BNE         LOOP4_2
                MOVE.L      #11,D3 1. ENTER ALARM TEMPERATURE MENU
                BRA         END4
LOOP4_2         CMP.L       #'2',D5
                BNE         RUNAGIN4
                MOVE.L      #8,D3  2. ADMINISTRATOR MAIN MENU
                BRA         END4
TIMEOUT4        MOVE.L      #20,D3 THE STATUS MENU
```

```
                BCLR        #2,D4  SETS THE THIRD BIT IN D4 LOW
END4            RTS
**************************************END

*11  (4) NUMERIC MENU "ENTER_ALARM_TEMPERATURE"
*FINISHED
**************************************
P_SETALARM_A    MOVE.B      #$F,$900001     DISPLAY ON AND CURSOR ON
                BSR         WAIT
                BSR         RESET_LCD       Administrator
                LEA         ADMIN,A6        Enter
                BSR         PRINT_LINE      Alarm temp.
                BSR         SECOND
                LEA         ENTER,A6
                BSR         PRINT_LINE
                BSR         THIRD
                LEA         ALARM_TEMP,A6
                BSR         PRINT_LINE
                BSR         FOURTH
**************************************ENTRY
                BSR         WAIT
                MOVE.L      #0,D2
                MOVE.B      MIN_TEMP,MIN_TEMPOLD THE MIN ALARM TEMPERATURE IS SAVED
                MOVE.B      MAX_TEMP,MAX_TEMPOLD THE MAX ALARM TEMPERATURE IS SAVED
                LEA         MIN_TEMP,A0
                LEA         MIN_TEMP_A,A1
RUNAGIN33       MOVE.L      #0,D0
RUN33           BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BNE         ACTIVE33
                ADD.L       #1,D0
                CMP.L       #TIME_OUT,D0
                BEQ         TIMEOUT33
                BRA         RUN33
ACTIVE33        CMP.L       #'A',D5
                BEQ         RUNAGIN33
                MOVE.B      D5,$900003      PLACE THE ENTRY ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      D5,(A1)+        SAVE THE INPUT  AS ASCII
                ADD.L       #1,D2

                CMP.L       #1,D2           * CONVERT TO NORMAL INTEGER
                BNE         CONVERT33A      *
                MULU        #10,D1
                BRA         CONVERT33B
CONVERT33A      CMP.L       #3,D2
                BNE         CONVERT33
                MULU        #10,D1          *
                BRA         CONVERT33B
CONVERT33       ADD.B       (A0),D1         *
CONVERT33B      MOVE.B      D1,(A0)         *SAVE THE INPUT
            ****************************
                CMP.L       #2,D2
                BEQ         PRINT1
                CMP.L       #4,D2
                BEQ         VAL33           *DO VALIDATION
                BRA         RUNAGIN33
TIMEOUT33       MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                MOVE.B      MIN_TEMPOLD,MIN_TEMP THE VAULE OF MIN AND MAX RETURNS TO THE VAULE SET BEFORE
                MOVE.B      MAX_TEMPOLD,MAX_TEMP
                MOVE.L      #20,D3
                BCLR        #2,D4           SETS THE THIRD BIT IN D4 LOW
                RTS
```

```
VAL33       MOVE.B    #$C,$900001     DISPLAY ON AND CURSOR OFF
            BSR       WAIT
            LEA       MAX_TEMP,A0
            LEA       MIN_TEMP,A1
            LEA       TEMPX_D,A3
            MOVE.B    (A1),D2         *D2 IS #4 AT THIS TIME, SHOULD OVERWRITE CORRECTLY LIKE THIS
            MOVE.B    (A0)+,D0        *
            MOVE.B    (A3)+,D1        *
          ****************************
            CMP.B     D0,D1
            BGE       INVAL33
            CMP.B     D1,D2
            BGE       INVAL33
          ****************************
            CMP.B     D0,D2
            BGE       INVAL33         *IF VALID RANGE, CONTINUE ELSE FORCE USER TO REDO IT
            CMP.B     #80,D0          * <-------------------MAX TEMPERATURE
            BHI       INVAL33         *
            CMP.B     #0,D2           * <-------------------MIN TEMPERATURE
            BLT       INVAL33         *
            BRA       END33           *
INVAL33     MOVE.L    #11,D3
            BSR       WAIT_OUT
            RTS
END33       BSR       WAIT_OUT
            MOVE.L    #8,D3
            RTS
PRINT1      MOVE.B    #$2D,$900003    PRINT (-) ON DISPLAY AFTER FIRST ENTRY
            BSR       WAITBF
            LEA       MAX_TEMP,A0
            LEA       MAX_TEMP_A,A1
            BRA       RUNAGIN33
*************************************END

*12 JUMP MENU "SET_TEMPERATURE_ADMIN"
*ALMOST FINISHED
****************************************
P_TEMP_A    MOVE.B    #$C,$900001     DISPLAY ON AND CURSOR OFF
            BSR       WAIT            MENU SETUP
            BSR       RESET_LCD       ****************
            LEA       TEMP_NULL,A6    *Temperature    *
            BSR       PRINT_LINE      *Current        *
            BSR       SECOND          *Set           1*
            LEA       CURRENT,A6      *Back          2*
            BSR       PRINT_LINE      ****************
            BSR       THIRD
            LEA       SET_ALARM,A6
            BSR       PRINT_LINE
            BSR       FOURTH
            LEA       BACK_2,A6
            BSR       PRINT_LINE
            *UPDATE THE VARIABLES
            MOVE.B    #$CC,$900001    LOCATION COMMAND
            BSR       WAIT
            LEA       TEMPX,A6
            MOVE.B    (A6)+,$900003     PRINT THE FIRST DIGIT OF THE SET TEMPERATURE ON THE DISPLAY
            BSR       WAITBF
            MOVE.B    (A6),$900003    PRINT THE SECOND DIGIT OF THE SET TEMPERATURE ON THE DISPLAY
            BSR       WAITBF
            MOVE.B    #$DF,$900003    PRINT (CIRC) ON THE DISPLAY
            BSR       WAITBF
            MOVE.B    #$43,$900003    PRINT ( C ) ON THE DISPLAY
****************************************
RUNAGIN11   MOVE.L    #0,D0           THE COUNTER IS RESET
```

```
L11             BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BNE         ACTIVE_11
                ADD.L       #1,D0           TIMEOUT COUNTER INCREMENTED
                CMP.L       #TIME_OUT,D0    TIME OUT IF D0 = 30                    *>REMEMBER TO CHANGE
                BEQ         TIMEOUT11
                BRA         L11
ACTIVE_11       CMP.L       #'A',D5
                BEQ         RUNAGIN11
                CMP.L       #'1',D5
                BNE         LOOP11_2
                MOVE.L      #13,D3          1. THE ADMIN ENTER SET TEMPERATURE
                BRA         END11
LOOP11_2        CMP.L       #'2',D5
                BNE         RUNAGIN11
                MOVE.L      #8,D3           2. RETURN TO ADMIN MAIN MENU
                BRA         END11
TIMEOUT11       MOVE.L      #20,D3          THE STATUS MENU
                BCLR        #2,D4           SETS THE THIRD BIT IN D4 LOW
END11           RTS
************************************END


*13 (2) NUMERIC  MENU "ENTER␣SET␣TEMPERATURE␣ADMIN"
*FINISHED
****************************************
P_SETTEMP_A     MOVE.B      #$F,$900001     DISPLAY ON AND CURSOR ON
                BSR         WAIT
                BSR         RESET_LCD       ********************
                LEA         SET_TEMPM,A6    *Set temp          *
                BSR         PRINT_LINE      *Enter             *
                BSR         SECOND          *Temperature       *
                LEA         ENTER,A6        *                  *
                BSR         PRINT_LINE      ********************
                BSR         THIRD
                LEA         TEMP_LONG,A6
                BSR         PRINT_LINE
****************************************
                BSR         FOURTH
                MOVE.L      #0,D2           COUNTER RESET
                MOVE.B      TEMPX_D,TEMP_OLD THE OLD SET TEMPERATURE VAULE IS SAVED
                MOVE.W      TEMPX,TEMP_OLD_A
                LEA         TEMPX,A0        POINT TO THE SET TEMPERATURE SPACE
RUNAGIN18       MOVE.L      #0,D0           COUNTER RESET
RUN18           BSR         KEYPAD_C
                CMP.L       #'X',D5
                BNE         ACTIVE18
                ADD.L       #1,D0
                CMP.L       #TIME_OUT,D0    TIME OUT IF D0 = 30
                BEQ         TIMEOUT18
                BRA         RUN18
ACTIVE18        CMP.L       #'A',D5
                BEQ         RUNAGIN18
                MOVE.B      D5,$900003      PLACE THE ENTRY ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      D5,(A0)+        SAVE THE INPUT (NOTE NOT AS ACSII)
                ADD.L       #1,D2
                CMP.L       #1,D2
                BEQ         MUL18
                ADD.B       D1,TEMPX_D
                CMP.L       #2,D2
                BEQ         END18
                BRA         RUNAGIN18
```

```
TIMEOUT18       MOVE.B      #$C,$900001       DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                MOVE.W      TEMP_OLD,TEMPX_D THE SET TEMPERATURE RETURNS TO THE OLD VAULE
                MOVE.L      #20,D3
                BCLR        #2,D4             SETS THE THIRD BIT IN D4 LOW
                RTS
END18           MOVE.B      #$C,$900001       DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                MOVE.W      TEMPX,D2          D2 IS #2 BEFORE THE MOVE   STOP_1
                CMP.W       #$3430,D2
                BLE         VAL40             IF LESS OR EQUAL TO 40 BRANCH TO VAL40
                BRA         ERROR             STOP IF BIGGER THAT 40
VAL40           CMP.W       #$3030,D2
                BHI         VALMIN            IF BIGGER TO 0 BRANCH TO COMPARE FOR MAX TEMPERATURE
                BRA         ERROR
VALMIN          CMP.W       MIN_TEMP_A,D2
                BHI         VALMAX
                BRA         ERROR
VALMAX          CMP.W       MAX_TEMP_A,D2
                BLT         VALID_ENTRY
                BRA         ERROR
ERROR           MOVE.B      TEMP_OLD,TEMPX_D
                MOVE.W      TEMP_OLD_A,TEMPX
                MOVE.L      #13,D3
                RTS
VALID_ENTRY     BSR         WAIT_OUT          STOP_1
                MOVE.L      #8,D3             RETURN TO ADMIN MAIN MENU
                RTS
MUL18           MULU        #10,D1
                MOVE.B      D1,TEMPX_D
                BRA         RUNAGIN18
*************************************END

*14 (8) NUMERIC MENU "USER_PHONENUMBER"
*FINISHED
****************************************
P_USERREG1      CMP.L       #0,USER_NR        SEE IF THERE ALREADY IS A USER PROIELE
                BNE         END13             END IF ENTRY PRESENT
                MOVE.B      #$F,$900001       DISPLAY ON AND CURSOR ON
                BSR         WAIT              MENU SETUP
                BSR         RESET_LCD         ****************
                LEA         USER_TOP,A6       *User           *
                BSR         PRINT_LINE        *Enter          *
                BSR         SECOND            *Phone number   *
                LEA         ENTER,A6          *+45            *
                BSR         PRINT_LINE        ****************
                BSR         THIRD
                LEA         PHONE_NR1,A6
                BSR         PRINT_LINE
                BSR         FOURTH
                LEA         PHONE_NR2,A6
                BSR         PRINT_LINE
****************************************
                MOVE.B      #$D3,$900001
                BSR         WAIT
                LEA         USER_NR,A6        A1 POINTS TO THE USER PHONE NUMBER SPACE
                MOVE.L      #0,D2
RUNAGIN13       MOVE.L      #0,D0
RUN13           BSR         KEYPAD_C          THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BNE         ACTIVE13
                ADD.L       #1,D0
                CMP.L       #TIME_OUT,D0
                BEQ         TIMEOUT13
```

```
                BRA         RUN13
ACTIVE13        CMP.L       #'A',D5
                BEQ         RUNAGIN13
                MOVE.B      D5,$900003      PLACE THE ENTRY ON THE DISPLAY
                MOVE.B      D5,(A6)+        SAVE THE INPUT
                ADD.L       #1,D2
                CMP         #8,D2
                BNE         RUNAGIN13
                BSR         WAIT_OUT
                MOVE.L      #15,D3          THE NEXT MENU IS THE "DAYS_ACTIVE_MENU"
                RTS
END13           MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                BSR         WAIT_OUT
                MOVE.L      #16,D3          THE USER MAIN MENU
                RTS
TIMEOUT13       MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                MOVE.L      #20,D3          THE STATUS MENU
                MOVE.B      #0,USER_NR
                BCLR        #2,D4           SETS THE THIRD BIT IN D4 LOW
                RTS
*************************************END

*15  (2) NUMERIC  MENU "NEW_USER_DAYS_ACTIVE"
*FINISHED
****************************************
P_USERREG_2     MOVE.B      #$F,$900001     DISPLAY ON AND CURSOR ON
                BSR         WAIT            MENU SETUP
                BSR         RESET_LCD       ********************
                LEA         NEW_USER,A6     *NEW USER          *
                BSR         PRINT_LINE      *DAYS ACTIVE       *
                BSR         SECOND          *                  *
                LEA         ENTER,A6        *                  *
                BSR         PRINT_LINE      ********************
                BSR         THIRD
                LEA         DAYSACTIVE,A6
                BSR         PRINT_LINE
****************************************
                BSR         FOURTH
                MOVE.L      #0,D2
                LEA         DAYS_ACTIVE,A0  POINT TO THE DAYS ACTIVE SPACE
RUNAGIN8        MOVE.L      #0,D0           COUNTER RESET
RUN8            BSR         KEYPAD_C
                CMP.L       #'X',D5
                BNE         ACTIVE8
                ADD.L       #1,D0
                CMP.L       #TIME_OUT,D0    TIME OUT IF D0 = 30
                BEQ         TIMEOUT8
                BRA         RUN8
ACTIVE8         CMP.L       #'A',D5
                BEQ         RUNAGIN8
                MOVE.B      D5,$900003      PLACE THE ENTRY ON THE DISPLAY
                BSR         WAITBF

                ADD.L       #1,D2
                ADD.B       (A0),D1
                CMP.L       #1,D2
                BNE         JUMP8
                MULU        #10,D1
JUMP8           MOVE.B      D1,(A0)             SAVE THE INPUT

                CMP.L       #2,D2
                BEQ         END8
```

```
                BRA         RUNAGIN8
TIMEOUT8        MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                MOVE.L      #20,D3          THE STATUS MENU
                BCLR        #2,D4           SETS THE THIRD BIT IN D4 LOW
                RTS
END8            MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                BSR         WAIT_OUT
                MOVE.L      #16,D3          THE USER MAIN MENU
                RTS
*************************************END

*16 JUMP MENU "USER_MAIN_MENU"
*************************************
P_USERMAIN      MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT            MENU SETUP
                BSR         RESET_LCD       ****************
                LEA         USER_TOP,A6     *User            *
                BSR         PRINT_LINE      *Temperature   1*
                BSR         SECOND          *Control       2*
                LEA         TEMP_LONG,A6    *                *
                BSR         PRINT_LINE      ****************
                BSR         THIRD
                LEA         CONTROL2,A6
                BSR         PRINT_LINE
*************************************
RUNAGIN2        MOVE.L      #0,D0           THE COUNTER IS RESET
L2              BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BNE         ACTIVE_2
                ADD.L       #1,D0           TIMEOUT COUNTER INCREMENTED
                CMP.L       #TIME_OUT,D0    TIME OUT IF D0 = 30
                BEQ         TIMEOUT2
                BRA         L2
ACTIVE_2        CMP.L       #'A',D5
                BEQ         RUNAGIN2
                CMP.L       #'1',D5
                BNE         LOOP2_2
                MOVE.L      #18,D3          1. THE USER TEMPERATURE MENU18
                BRA         END2
LOOP2_2         CMP.L       #'2',D5
                BNE         RUNAGIN2
                MOVE.L      #17,D3          2. THE USER CONTROL MENU17
                BRA         END2
TIMEOUT2        MOVE.L      #20,D3          THE STATUS MENU IS RAN NEXT TIME MAIN-MENU IS RUN
                BCLR        #2,D4           SETS THE THIRD BIT IN D4 LOW
END2            RTS
*************************************END

*17 JUMP MENU CONTROL MENU USER
*FINISHED
*************************************
P_CONTROL_U     MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT            MENU SETUP
                BSR         RESET_LCD       ****************
                LEA         CONTROL,A6      *CONTROL         *
                BSR         PRINT_LINE      *Turn ON       1*
                BSR         SECOND          *Turn OFF      2*
                LEA         TURN_ON,A6      *Back          3*
                BSR         PRINT_LINE      ****************
                BSR         THIRD
                LEA         TURN_OFF,A6
                BSR         PRINT_LINE
```

```
                BSR         FOURTH
                LEA         BACK_3,A6
                BSR         PRINT_LINE
*************************************
RUNAGIN6        MOVE.L      #0,D0           THE COUNTER IS RESET
L6              BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BNE         ACTIVE_6
                ADD.L       #1,D0           TIMEOUT COUNTER INCREMENTED
                CMP.L       #TIME_OUT,D0    TIME OUT IF D0 = 30
                BEQ         TIMEOUT6
                BRA         L6
ACTIVE_6        CMP.L       #'1',D5         1. TURN ON
                BNE         LOOP6_2
                BSET.L      #0,D4           SETS THE FIRST BIT IN D4 HIGH I.E. THE SYSTEM IS IN CONTROL MODE
                BRA         END6
LOOP6_2         CMP.L       #'2',D5         2. TURN OFF
                BNE         LOOP6_3
                BCLR.L      #0,D4           SETS THE FIRST BIT IN D4 LOW  I.E. THE SYSTEM IS IN MONITORING MODE
                BRA         END6
LOOP6_3         CMP.L       #'3',D5         3. BACK
                BNE         RUNAGIN6
                BRA         END6
TIMEOUT6        MOVE.L      #20,D3          STATUS MENU
                BCLR        #2,D4           SETS THE THIRD BIT IN D4 LOW
                RTS
END6            MOVE.L      #16,D3          RETURN TO USER MAIN MENU
                RTS
*************************************END

*18 JUMP MENU "SET␣TEMPERATURE␣USER"
*************************************
P_TEMP_U        MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT            MENU SETUP
                BSR         RESET_LCD       ****************
                LEA         TEMP_NULL,A6    *Temperature    *
                BSR         PRINT_LINE      *Current        *
                BSR         SECOND          *Set           1*
                LEA         CURRENT,A6      *Back          2*
                BSR         PRINT_LINE      ****************
                BSR         THIRD
                LEA         SET_ALARM,A6
                BSR         PRINT_LINE
                BSR         FOURTH
                LEA         BACK_2,A6
                BSR         PRINT_LINE
                *UPDATE THE VARIABLES
                MOVE.B      #$CC,$900001    LOCATION COMMAND
                BSR         WAIT
                LEA         TEMPX,A6
                MOVE.B      (A6)+,$900003     PRINT THE FIRST DIGIT OF THE SET TEMPERATURE ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      (A6),$900003    PRINT THE SECOND DIGIT OF THE SET TEMPERATURE ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      #$DF,$900003    PRINT (CIRC) ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      #$43,$900003    PRINT ( C ) ON THE DISPLAY
*************************************
RUNAGIN7        MOVE.L      #0,D0           THE COUNTER IS RESET
L7              BSR         KEYPAD_C        THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BNE         ACTIVE_7
                ADD.L       #1,D0           TIMEOUT COUNTER INCREMENTED
                CMP.L       #TIME_OUT,D0    TIME OUT IF D0 = 30                    *>REMEMBER TO CHANGE
```

```
                BEQ         TIMEOUT7
                BRA         L7
ACTIVE_7        CMP.L       #'A',D5
                BEQ         RUNAGIN7
                CMP.L       #'1',D5
                BNE         LOOP7_2
                MOVE.L      #19,D3          1. THE USER ENTER SET TEMPERATURE
                BRA         END7
LOOP7_2         CMP.L       #'2',D5
                BNE         RUNAGIN7
                MOVE.L      #16,D3          2. RETURN TO USER MAIN MENU
                BRA         END7
TIMEOUT7        MOVE.L      #20,D3          THE STATUS MENU
                BCLR        #2,D4           SETS THE THIRD BIT IN D4 LOW
END7            BSR         WAIT_OUT
                RTS
*************************************END

*19 (2) NUMERIC  MENU "ENTER␣SET␣TEMPERATURE␣USER"
*FINISHED
***************************************
P_SETTEMP_U     MOVE.B      #$F,$900001     DISPLAY ON AND CURSOR ON
                BSR         WAIT
                BSR         RESET_LCD       ******************
                LEA         SET_TEMPM,A6    *Set temp        *
                BSR         PRINT_LINE      *Enter           *
                BSR         SECOND          *Temperature     *
                LEA         ENTER,A6        *                *
                BSR         PRINT_LINE      ******************
                BSR         THIRD
                LEA         TEMP_NULL,A6
                BSR         PRINT_LINE
***************************************
                BSR         FOURTH
                MOVE.L      #0,D2           COUNTER RESET
                MOVE.B      TEMPX_D,TEMP_OLD THE OLD SET TEMPERATURE VAULE IS SAVED
                MOVE.W      TEMPX,TEMP_OLD_A
                LEA         TEMPX,A0        POINT TO THE SET TEMPERATURE SPACE
RUNAGIN32       MOVE.L      #0,D0           COUNTER RESET
RUN32           BSR         KEYPAD_C
                CMP.L       #'X',D5
                BNE         ACTIVE32
                ADD.L       #1,D0
                CMP.L       #TIME_OUT,D0    TIME OUT IF D0 = 30
                BEQ         TIMEOUT32
                BRA         RUN32
ACTIVE32        CMP.L       #'A',D5
                BEQ         RUNAGIN32
                MOVE.B      D5,$900003      PLACE THE ENTRY ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      D5,(A0)+        SAVE THE INPUT (NOTE NOT AS ACSII)
                ADD.L       #1,D2
                CMP.L       #1,D2
                BEQ         MUL32
                ADD.B       D1,TEMPX_D
                CMP.L       #2,D2
                BEQ         END32
                BRA         RUNAGIN32
TIMEOUT32       MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                MOVE.W      TEMP_OLD,TEMPX_D THE SET TEMPERATURE RETURNS TO THE OLD VAULE
                MOVE.L      #20,D3
                BCLR        #2,D4           SETS THE THIRD BIT IN D4 LOW
                RTS
```

```
END32           MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                MOVE.W      TEMPX,D2        D2 IS #2 BEFORE THE MOVE   STOP_1
                CMP.W       #$3430,D2
                BLE         VAL32           IF LESS OR EQUAL TO 40 BRANCH TO VAL40
                BRA         ERROR32         STOP IF BIGGER THAT 40
VAL32           CMP.W       #$3030,D2
                BHI         VALMIN32        IF BIGGER TO 0 BRANCH TO COMPARE FOR MAX TEMPERATURE
                BRA         ERROR32
VALMIN32        CMP.W       MIN_TEMP_A,D2
                BHI         VALMAX32
                BRA         ERROR32
VALMAX32        CMP.W       MAX_TEMP_A,D2
                BLT         VALID_ENTRY32
                BRA         ERROR32
ERROR32         MOVE.B      TEMP_OLD,TEMPX_D
                MOVE.W      TEMP_OLD_A,TEMPX
                MOVE.L      #20,D3
                RTS
VALID_ENTRY32   BSR         WAIT_OUT        STOP_1
                MOVE.L      #16,D3          RETURN TO USER MAIN MENU
                RTS
MUL32           MULU        #10,D1
                MOVE.B      D1,TEMPX_D
                BRA         RUNAGIN32
*************************************END

*20 NORMAL OPERATIONS MENU "STATUS"
*FINISHED
****************************************
P_STATUS        MOVE.B      #$C,$900001     DISPLAY ON AND CURSOR OFF
                BSR         WAIT
                BTST.L      #2,D4           COMPARE THE FIRST BIT IN D4 TO 0
                BNE         UPDATE
                BSR         RESET_LCD       *************************
                LEA         CTEMP,A6        *C. temp     +13.4(circ)C*
                BSR         PRINT_LINE      *Control                ON*
                BSR         SECOND          *Time                12:32*
                LEA         CONTROL,A6      *GSM  ON                OFF*
                BSR         PRINT_LINE      *************************
                BSR         THIRD
                LEA         TIME,A6
                BSR         PRINT_LINE
                BSR         FOURTH
                LEA         GSM,A6
                BSR         PRINT_LINE
****************************************
                *UPDATE THE TEMPERATURE
UPDATE          NOP
                MOVE.B      #$89,$900001    LOCATION CAMMAND; FIRST LINE END - 7
                BSR         WAIT
                *TEMPERATURE LOCATED IN D7
                ROL.L       #8,D7
                MOVE.B      D7,$900003      PRINT THE FIRST DIGIT ON THE DISPLAY
                BSR         WAITBF
                ROL.L       #8,D7
                MOVE.B      D7,$900003      PRINT THE SECOND DIGIT ON THE DISPLAY
                BSR         WAITBF
                ROL.L       #8,D7
                MOVE.B      D7,$900003      PRINT
                BSR         WAITBF
                MOVE.B      #$2E,$900003    PRINT ( . )  ON THE DISPLAY
                BSR         WAITBF
                ROL.L       #8,D7
```

```
                MOVE.B      D7,$900003        PRINT THE FOURTH DIGIT ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      #$DF,$900003      PRINT (CIRC) ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      #$43,$900003      PRINT "C" ON THE DISPLAY
                BSR         WAITBF
*****************************************
                *UPDATE THE STATE OF THE SYSTEM
                MOVE.B      #$CD,$900001      LOCATION COMMAND; SECOND LINE END - 3.
                BSR         WAIT
                BSR         ONOFF
                BSR         PRINT_LINE        PRINT THE STATE OF THE SYSTEM (CONTROL ON/OFF) ON THE DISPLAY
*****************************************
                *UPDATE THE SYSTEM TIME
                MOVE.B      #$9B,$900001      LOCATION COMMAND; THIRD LINE END - 4.
                BSR         WAIT
                *SYSTEM TIME LOCATED IN D6 [MM:HH]
                ROL.L       #8,D6
                MOVE.B      D6,$900003        PRINT THE FIRST DIGIT ON THE DISPLAY
                BSR         WAITBF
                ROL.L       #8,D6
                MOVE.B      D6,$900003        PRINT THE SECOND DIGIT ON THE DISPLAY
                BSR         WAITBF
                MOVE.B      #$3A,$900003      PRINT ( : ) ON DISPLAY
                BSR         WAITBF
                ROL.L       #8,D6
                MOVE.B      D6,$900003        PRINT THE THIRD DIGIT ON THE DISPLAY
                BSR         WAITBF
                ROL.L       #8,D6
                MOVE.B      D6,$900003        PRINT THE FOURTH DIGIT ON THE DISPLAY
                BSR         WAITBF
*****************************************
                *UPDATE THE STATE OF THE GSM-MODULE
                MOVE.B      #$D9,$900001      LOCATION CAMMAND; FOURTH LINE END - 7.
                BSR         WAIT
                BSR         GSM_ONOFF
                BSR         PRINT_LINE        PRINT THE STATE OF THE GSM-MODULE ON THE DISPLAY
                BSET        #2,D4             SETS THE THIRD BIT IN D4 HIGH (NOW ONLY THE VARIABLES ARE UPDATED)
                BSR         KEYPAD_C          THE KEYPAD ENTRY FUNCTION IS RAN
                CMP.L       #'X',D5
                BEQ         INACTIVE
                MOVE.L      #5,D3             FORCED TO USER MODE MENU IF ENTRY IS MADE
INACTIVE        RTS
*****************************************END
```

I.2 Subroutines

```
*START_SUBS ORG         $47000
*********************************************
*SUBROUTINES USED IN THE MENU  FUNCTIONS    *
*********************************************

* INITIALIZE THE LCD
*********************************************
INIT_LCD    MOVE.B      #$38,$900001
            BSR         WAIT
            MOVE.B      #$38,$900001
            BSR         WAIT
            MOVE.B      #$38,$900001      FUNCTION SET
            BSR         WAIT
            MOVE.B      #$6,$900001       ENTRY MODE SET
            BSR         WAIT
```

```
            MOVE.B      #$F,$900001     DISPLAY ON AND CURSOR ON (BLINKING)
            BSR         WAIT
            MOVE.B      #$1,$900001     DISPLAY CLEAR
            BSR         WAIT

* DEFINE DEFAULT SETTINGS
 ****************************************
            MOVE.B      #0,MIN_TEMP         THE DEFAULT MIN ALARM TEMP IN DECIMAL
            MOVE.B      #35,MAX_TEMP        THE DEFAULT MAX ALARM TEMP IN DECIMAL
            MOVE.W      #$3030,MIN_TEMP_A   THE DEFAULT MIN ALARM TEMP IN ASCI
            MOVE.W      #$3335,MAX_TEMP_A   THE DEFAULT MIN ALARM TEMP IN ASCI
            MOVE.B      #0,DAYS_ACTIVE
            MOVE.B      #1,DAYS_ACTIVE_C
            MOVE.B      #'0',ADMIN_NR
            MOVE.B      #'0',ADMIN_NR1
            MOVE.L      #0,USER_NR
            MOVE.B      #10,ENTRY
            MOVE.B      #16,ENTRY
            MOVE.L      #$2B323030,D7
            MOVE.L      #2,D3               THE DEFAULT STARTUP MENU NUMBER
            MOVE.W      #$3231,TEMPX        THE DEFAULT SET TEMPERATURE 21(CIRC)C
            MOVE.L      #$2B303030,D7
            BCLR.L      #0,D4
            BCLR.L      #1,D4
            MOVE.B      #21,TEMPX_D         THE DEFAULT SET TEMPERATURE IN DECIMAL (21)
            MOVE.B      #0,MIN_TEMP
            MOVE.B      #35,MAX_TEMP
            MOVE.B      #30,TEMP_C_D
            MOVE.W      #0,ALARM_COUNT1     THE ALARM COUNTER IS SET TO 0 - THE FIRST MESSAGE WILL BE SENT AS SO
            ADDI.B      #0,$D00001          ;Sluk kontrolenhed
            MOVE.B      #$FE,CLKCOUNTER
            MOVE.B      #$7E,GSMCOUNTER
            MOVE.B      #0,TEMPCOUNTER
            MOVE.B      #0,ALARMCOUNTER
            RTS
*****************************************END

* DISPLAY CLEAR
*****************************************
RESET_LCD   BSR         WAIT
            MOVE.B      #$1,$900001         DISPLAY CLEAR
            BSR         WAIT
            RTS

* THE STANDARD PRINTING-RUTINE
*****************************************
PRINT_LINE  BSR         WAITBF
L69         CMP.B       #0,(A6)             CONTROL IF THE COMMAND IS AT THE END OF THE TEXT STRING
            BEQ         END_COM             END
            MOVE.B      (A6)+,$900003
            BSR         WAITBF
            BRA         L69                 REPEATS WHIT THE NEXT CHAR
END_COM     RTS

* WAIT ROUTINES
*****************************************
* COMMANDS
WAIT        MOVE.L      #0,D0
L12         ADD.L       #1,D0
   CMP.L        #3500,D0          (CREATES A WAIT THAT LAST 15.7 mS)
   BNE          L12
   RTS
*CHARACTERS
WAITBF      NOP
```

```
LL          BTST.B      #7,$900001          WAITS FOR THE BUSSY FLAG BIT TO GO LOW BEFORE MOVING ON
            BNE         LL
            RTS

*SMALL WAIT
WAIT2       MOVE.L      #0,D0               CREATES A WAIT PERIOD OF 0.228 mS
L15         ADD.L       #1,D0
            CMP.L       #50,D0
            BNE         L15
            RTS

*SMALL WAIT USED FOR THE TIMEOUT THE MENU IS VISEBLE FOR 1.03 SECONDS
WAIT_OUT    MOVE.L      #0,D0
L18         ADD.L       #1,D0
            CMP.L       #230000,D0
            BNE         L18
            RTS

* SECOND LINE FIRST CHARACTER JUMP
 *******************************************
SECOND      MOVE.B      #$C0,$900001
            BSR         WAIT2
            RTS

* THIRD LINE FIRST CHARACTER JUMP
 *******************************************
THIRD       MOVE.B      #$90,$900001
            BSR         WAIT2
            RTS
 *******************************************END

* FOURTH LINE FIRST CHARACTER JUMP
 *******************************************
FOURTH      MOVE.B      #$D0,$900001
            BSR         WAIT2
            RTS
 *******************************************END

*SUBROUTINES USED IN THE MENU  FUNCTIONS
 *******************************************


* CONTROL STATE ON/OFF
 *******************************************
ONOFF       BTST.L      #0,D4               COMPARE THE FIRST BIT IN D4 TO 0
            BNE         CON                 IF EQUAL TO 1 JUMP TO CON
            LEA         STATE_OFF,A6        A6 POINTS TO OFF
            RTS
CON         LEA         STATE_ON,A6         A6 POINTS TO ON
            RTS
*******************************************END

*GSM STATE ON/OFF
 *******************************************
GSM_ONOFF   BTST.L      #1,D4
            BNE         GSMON               IF EQUAL TO 1 JUMP TO ON
            LEA         OFFLINE,A6          A6 POINTS TO OFF
            RTS
GSMON       LEA         ONLINE,A6           A6 POINTS TO ON
   RTS
 *******************************************END

KEYPAD_C    NOP
            MOVE.L      #'X',D5
```

```
            CMP.B     #16,ENTRY
            BEQ       ENDKEYPAD
TEST1       CMP.B     #1,ENTRY
             BNE       TEST2
            MOVE.L    #'1',D5
            MOVE.B    #1,D1
            MOVE.B    #16,ENTRY
            BRA       ENDKEYPAD
TEST2       CMP.B     #2,ENTRY
            BNE       TEST3
            MOVE.L    #'2',D5
            MOVE.B    #2,D1
            MOVE.B    #16,ENTRY
            BRA       ENDKEYPAD
TEST3       CMP.B     #3,ENTRY
            BNE       TEST4
            MOVE.L    #'3',D5
            MOVE.B    #3,D1
            MOVE.B    #16,ENTRY
            BRA       ENDKEYPAD
TEST4       CMP.B     #5,ENTRY
            BNE       TEST5
            MOVE.L    #'4',D5
            MOVE.B    #4,D1
            MOVE.B    #16,ENTRY
            BRA       ENDKEYPAD
TEST5       CMP.B     #6,ENTRY
            BNE       TEST6
            MOVE.L    #'5',D5
            MOVE.B    #5,D1
            MOVE.B    #16,ENTRY
            BRA       ENDKEYPAD
TEST6       CMP.B     #7,ENTRY
            BNE       TEST7
            MOVE.L    #'6',D5
            MOVE.B    #6,D1
            MOVE.B    #16,ENTRY
            BRA       ENDKEYPAD
TEST7       CMP.B     #9,ENTRY
            BNE       TEST8
            MOVE.L    #'7',D5
            MOVE.B    #7,D1
            MOVE.B    #16,ENTRY
            BRA       ENDKEYPAD
TEST8       CMP.B     #10,ENTRY
            BNE       TEST9
            MOVE.L    #'8',D5
            MOVE.B    #8,D1
            MOVE.B    #16,ENTRY
            BRA       ENDKEYPAD
TEST9       CMP.B     #11,ENTRY
            BNE       TEST0
            MOVE.L    #'9',D5
            MOVE.B    #9,D1
            MOVE.B    #16,ENTRY
            BRA       ENDKEYPAD
TEST0       CMP.B     #14,ENTRY
            BNE       TESTENTER
            MOVE.L    #'0',D5
            MOVE.B    #0,D1
            MOVE.B    #16,ENTRY
TESTENTER   CMP.B     #13,ENTRY
            BNE       ENDKEYPAD
            MOVE.L    #'A',D5
```

```
                MOVE.B          #16,ENTRY
ENDKEYPAD       RTS

*****************************************END
```

Appendix J

ADC and Temperature software

```
******** Definitions ********
ADC_ADR     EQU         $C00001
ADC_CS      EQU         $47000

******** Main code ********
ADC         ADD.B       #1,TEMPCOUNTER
            MOVE.L      #0,D0
            MOVE.B      (ADC_ADR),D0
            ADD.W       TEMP_AV_D,D0
            MOVE.W      D0,TEMP_AV_D
            MOVE.B      TEMPCOUNTER,D1
            CMP.B       #$20,D1
            BNE         END_ADC
            MOVE.L      #0,D1
            DIVU        #$20,D0         Moves the content of addr $C00000 into D0
            BSR         CALC            Branch to subroutine CALC
            MOVE.W      #0,TEMP_AV_D
            MOVE.B      #0,TEMPCOUNTER
END_ADC     RTS


******** Calculation ********
CALC        ADD.W       #540,D0
            MULU.W      #15,D0
            DIVU.W      #32,D0
            CMP.W       #273,D0
            BLE         NEGATIVE
            BRA         POSITIVE
            NOP
POSITIVE    SUB.L       #273,D0
            MOVE.B      D0,TEMP_C_D
            MOVE.B      #'+',D2
            BRA         TO_10
            NOP
NEGATIVE    MOVE.W      #273,D1
            SUB.W       D0,D1
            MOVE.W      D1,D0
            MOVE.B      #'-',D2
            BRA         TO_10
** This does integer division, at 10s and 1s.
** Calculates 10s
TO_10       MOVE.W      D0,D1
            DIVU.W      #10,D1
```

```
            ROL.L       #8,D2
            MOVE.B      D1,D2
            SWAP.W      D1
** Calculates 1s
            AND.L       #$FFFF,D1
            ROL.L       #8,D2
            MOVE.B      D1,D2
            SWAP.W      D1
** Check for .0 or .5
            SWAP.W D0
            ANDI.L      #$FFFF,D0
            DIVU.W      #16,D0
            MULU.W      #5,D0
            ROL.L       #8,D2
            MOVE.B      D0,D2
            ADD.L       #$303030,D2
            MOVE.L      D2,D7
            RTS


*TEMPCOUNTER    DS.B  1
*TEMP_AV_D  DS.W  1
```

Appendix K

Alarm software

```
ALARM_UP      ADD.B       #1,ALARMCOUNTER    ADD 1 TO CURRENT ALARM COUNT
              CMP.B       #$60,ALARMCOUNTER
              BEQ         ALARM_UP_6         IF COUNTER IS $60 / 96, GO TO ALARM_UP_6
              RTS                            ELSE RETURN TO MAIN LOOP

ALARM_UP_6    MOVE.L      #0,D0              CLEAR REGISTERS
              MOVE.L      #0,D1
              MOVE.L      #0,D2
              MOVE.B      #0,ALARMCOUNTER
              MOVE.W      MIN_TEMP_A,D1    LOAD MINIMUM ALARM TEMPERATURE
              MOVE.W      MAX_TEMP_A,D2    LOAD MAXIMUM ALARM TEMPERATURE
              ROR.L       #8,D7              ROLL TO THE INTEGER VALUE OF THE TEMPERATURE
              MOVE.W      D7,D0              LOAD IT INTO D0
              ROL.L       #8,D7              ROLL D7 BACK
              CMP.W       D0,D1  CHECK IF MIN.  TEMPERATURE IS REACHED
              BGT         ALARM_UP_1         IF NOT, SET OFF ALARM
              CMP.W       D0,D2      CHECK IF MAX. TEMPERATURE IS REACHED
              BLT         ALARM_UP_1         IF SO, SET OFF ALARM
              MOVE.B      #0,ALARM_COUNT1    ELSE, RESET ALARM COUNTER 1 TO 0
              RTS                            AND RETURN TO MAIN LOOP

ALARM_UP_1    CMP.B       #0,ALARM_COUNT1    IS THIS THE FIRST ALARM?
              BNE         ALARM_UP_2         IF NOT, GO TO ALARM UPDATE 2
              MOVE.W      D6,ALARM_COUNT2    ELSE MOVE THE CURRENT MINUTE COUNT INTO ALARM COUNTER 2

ALARM_UP_2    CMP.W       ALARM_COUNT2,D6    COMPARE MINUTES IN ALARM COUNTER 2 TO THE CURRENT TIME
              BEQ         ALARM_UP_3         IN MINUTES, IF THEY MATCH, GO TO ALARM UPDATE 3
              RTS

ALARM_UP_3    CMP.B       #4,ALARM_COUNT1 CHECK IF 4 ALARMS HAS BEEN SENT YET
              BLE         ALARM_UP_4     IF NOT, GO SEND ANOTHER ONE
              RTS                            ELSE RETURN TO MAIN LOOP

ALARM_UP_4    MOVE.B      #14,SEND_MSG       LOAD ALARM MESSAGE
              BSR         SEND_SMS_A         SEND IT

              MOVE.B      ALARM_COUNT1,D0
              ADD.B       #1,D0
              MOVE.B      D0,ALARM_COUNT1    UPDATE ALARMS COUNTED TO PREV. VALUE +1

              MOVE.L      ALARM_COUNT2,D0    LOAD ALARM COUNTER 2 INTO D0
              CMP.W       #$3330,D0          SEE IF ITS 30 MINUTES OR MORE PAST ANY HOUR
              BGT         ALARM_UP_5         IF IT IS, GO TO ALARM UPDATE 5
              ADD.W       #$300,D0           ELSE ADD 30 MINUTES (ASCII) TO THE COUNTER
```

```
                MOVE.L          D0,ALARM_COUNT2
                RTS

ALARM_UP_5      SUB.W           #$300,D0                SUBTRACT 300 FROM ALARM COUNTER 2
                MOVE.L          D0,ALARM_COUNT2
                RTS
```

Appendix L

User termination software

```
************  PROGRAM  ************************
TERM_USER     LEA          USER_NR,A0
              CMP.L        #0,(A0)
              BEQ          TU_EXIT
***********************************************
              CMP.L        #'0030',D6          IS IT 00:00-00:30?
              BLT          TU_NEWDAY           IF YES, CHECK IF ITS A NEW DAY
              MOVE.B       #0,DAYS_COUNT       ELSE SET DAY COUNTED TO 0
TU_NEWDAY     CMP.L        #'0030',D6          IS IT PAST 00:30
              BGE          TU_GO               IF SO, GO CHECK IF USER NEEDS TO BE TERMINATED
              CMP.B        #1,DAYS_COUNT       ELSE CHECK IF THE DAY HAS BEEN COUNTED YET
              BEQ          TU_GO               IF SO, GO CHECK IF USER SHOULD BE TERMINATED
              ADD.B        #1,DAYS_ACTIVE_C    ELSE COUNT THIS DAY
              MOVE.B       #1,DAYS_COUNT       SET DAY COUNTED TO 1
***********************************************
TU_GO         MOVE.B       DAYS_ACTIVE_C,D0    LOAD HOW MANY DAYS HAVE GONE
              LEA          DAYS_ACTIVE,A1      LOAD HAVE MANY DAYS USER WILL BE ACTIVE
              CMP.B        (A1),D0             HAS THE DAYS GONE BY?
              BGE          TU_TERM             IF SO, SEE IF ITS THE RIGHT TIME OF THE DAY
              BRA          TU_EXIT             ELSE EXIT TO MAIN LOOP
***********************************************
TU_TERM       CMP.L        #'1200',D6          TIME OF DAY TO TERMINATE USER, COMPARED TO CURRENT TIME
              BLT          TU_EXIT             IF ITS NOT TIME, EXIT TO MAIN LOOP
              MOVE.L       #0,USER_NR          ELSE, TERMINATE USER
              MOVE.B       #1,DAYS_ACTIVE_C    RETURN DAYS PASSED TO DEFAULT VALUE
TU_EXIT       RTS
```

Appendix M

Control unit software

```
CONTROL_UP  BTST      #0,D4          IS THE CONTROL UNIT ACTIVE?
            BNE       CMPTMP         IF IT IS, GO TO CMPTMP
            BRA       CONOFF         ELSE GO TO CONOFF

CMPTMP      CLR       D0             CLEAR REGISTERS
            CLR       D1
            MOVE.B    TEMPX_D,D0     LOAD USER SET TEMPERATURE INTO D0
            MOVE.B    TEMP_C_D,D1    LOAD CURRENT TEMPERATURE INTO D1
            ADDI.B    #1,D0          ADD 1 TO USER SET TEMPERATURE
            CMP.B     D1,D0          COMPARE USER TEMP +1 WITH CURRENT TEMPERATURE
            BLE       CONOFF         IF CURRENT TEMPERATURE IS HIGHER, GO TO CONOFF

            SUB.B     #2,D0          SUBTRACT 2 FROM D0, THIS GIVES USER TEMPERATURE -1
            CMP.B     D1,D0          COMPARE THIS TO CURRENT TEMPERATURE
            BGE       CONON          IF CURRENT IS LOWER OR EQUAL, GO TO CONON
            RTS                      IF NOT, RETURN TO MAIN LOOP
CONON       ADDI.B    #0,$D00003     TURN ON CONTROL UNIT
            RTS                      AND RETURN TO MAIN LOOP
CONOFF      ADDI.B    #0,$D00001     TURN OFF CONTROL UNIT
            RTS                      AND RETURN TO MAIN LOOP
```